AD-A049 303 HARRY DIAMOND LABS ADELPHI MD F/G 9/2
BDACS SOFTWARE.(U)
NOV 77 J C INGRAM MIPR-76628
UNCLASSIFIED HDL-TR-1831 NL

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) HDL-TR-1831	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) BDACS Software.	5. TYPE OF REPORT & PERIOD COVERED (9) Technical Report.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) John C. Ingram	8. CONTRACT OR GRANT NUMBER(s) (15) MIPR 76628
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Director Defense Nuclear Agency Washington, DC 20305	12. REPORT DATE November 1977
	13. NUMBER OF PAGES 127
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 122p.	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

(16) L37EAXY (17) X910

DDC RECEIVED FEB 2 1978 B

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

HDL Project: E436E2
This research was sponsored by the Defense Nuclear Agency under subtask L37EAXYX910, Work Unit 09, Binary Data Acquisition and Control System (BDACS).

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Binary data acquisition
Computer control system software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Binary Data Acquisition and Control System (BDACS) has undergone extensive hardware augmentation to provide additional high-speed and low-speed monitoring. In a parallel effort, the software control package for the BDACS has undergone a similar major modification to provide the necessary control functions for the extended hardware. Moreover, the restructed software

→ next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

1 SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

163 050 4B

package has incorporated improvements in data acquisition and reduction efficiency and has provided a more detailed on-line printout of the data vector. Last, the new BDACS software package has been organized in a "structured program" modular format to allow for future expansion without complicated inter-dependencies among the several program modules.

CONTENTS

CONTENTS (Cont'd)

FIGURES

TABLE

1. INTRODUCTION AND HISTORICAL BACKGROUND

The Binary Data Acquisition and Control System (BDACS) is a computer-controlled general-purpose instrumentation system that retrieves and stores binary logic signals generated by a system undergoing some form of testing. Simultaneously, BDACS can transmit binary logic control signals to the test system in order to initiate a specific logic state of the system or to control the sequence of operations through several logic states.

BDACS [originally called the Communications Monitor and Control System (CMCS)] was developed under the Program for Electromagnetic Pulse Testing (PREMPT), jointly sponsored by the Defense Nuclear Agency (DNA) and the Defense Communications Agency (DCA). The primary objectives of PREMPT are:

(a) to evaluate the vulnerability/survivability of the WWMCCS/DCS Command, Control, and Communications systems (C^3) subjected to a high-altitude electromagnetic pulse (HEMP) environment by using a methodology that includes both testing and analysis;

(b) to provide hardening recommendations/fixes that insure an acceptable level of performance of the WWMCCS/DCS in a HEMP environment;

(c) to develop the analytical tools required to analyze the WWMCCS/DCS and other critical C^3 systems.

The initial phases of PREMPT included large-scale test programs to be performed on three telephone switch centers that form a part of the continental U.S. Automatic Voice Network (CONUS-AUTOVON). As part of the data requirements for these test programs, the logic states of the switch centers had to be monitored and controlled. The development of BDACS is a result of these requirements.

A block diagram of the BDACS hardware is shown in figure 1 (a to c). The hardware is centered around a Data General Corporation (DGC) Nova-1230 minicomputer processor and peripheral system. The computer includes a 16K core memory, Teletype (TTY) control unit, card-reader input unit, line-printer output unit (to allow real-time hard-copy output of BDACS results), a 9-track magnetic-tape drive (to allow archive storage of BDACS results), and a disk-cartridge drive to store both the background operating software system (supplied by DGC) and the foreground BDACS control software system that was developed concurrently with the hardware. The special-purpose interface board supplied by the original equipment manufacturer (OEM) provides the hardware link between the computer section and the binary signal interface/multiplexer and special-purpose control boards of BDACS. These special-purpose control

BLOCK DIAGRAMS OF BDACS

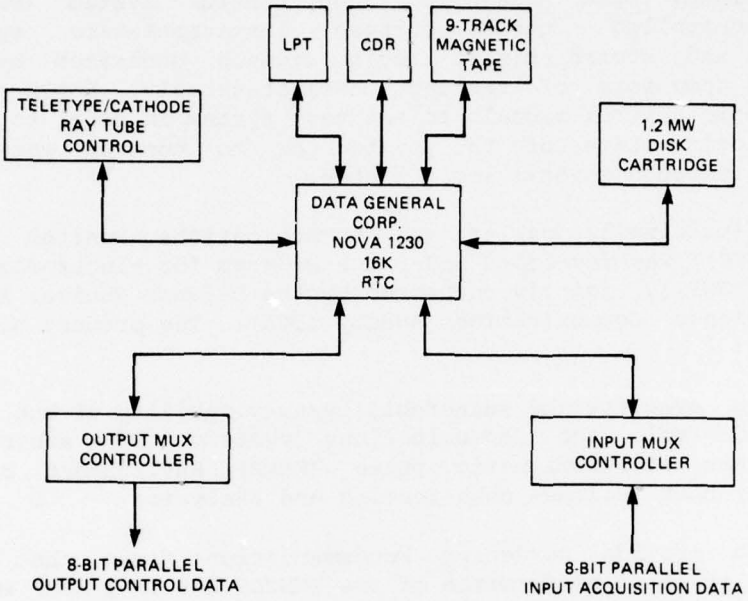

Figure 1(a). Central control and processor.

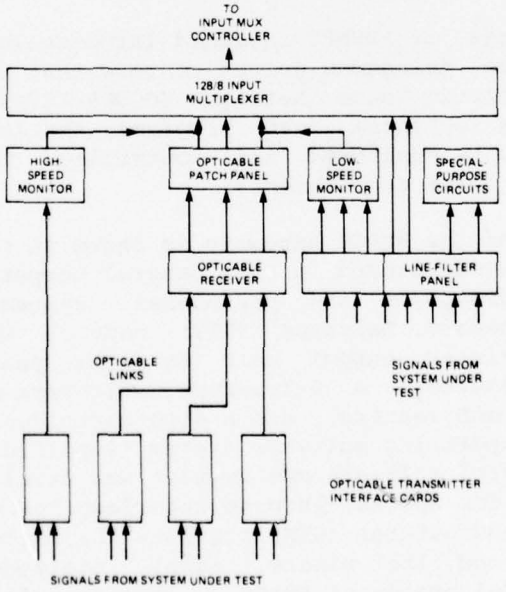

Figure 1(b). Input multiplexer unit.

BLOCK DIAGRAM OF BDACS

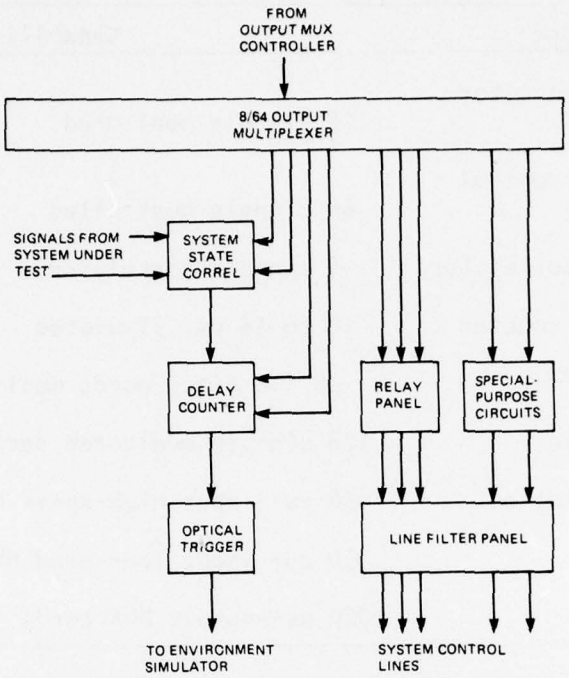

Figure 1(c). Output multiplexer unit.

boards include (1) a state correlator which generates a time-delayed trigger signal when the system under test enters a prescribed logic state (as defined by the parallel coincidence of up to eight logic signals) (2) a high-speed buffer (HSB) that allows very fast monitoring of sixteen binary signals at sample rates of two per microsecond, (3) a low-speed buffer (LSB) that allows a slow monitor and serial transfer of slowly changing logic signals, and (4) audible tone diallers and dial-tone detectors that are specific interface modules for the particular telephone switch centers which were tested under PREMPT. The present capabilities of BDACS are summarized in table I.

The heart of the BDACS consists of a software control program and several ancillary programs that provide the communications links between the BDACS operator and the system hardware. The remainder of this report describes the architecture and functions of several modules that comprise the BDACS software.

The initial BDACS software package was prepared by the OEM during the initial system development and was used during BDACS operations at Polk City, FL. During the BDACS refurbishing period in preparation of operations at Delta, UT, some minor modifications and improvements to

TABLE I. PRESENT BDACS CAPABILITIES

Feature	Capabilities
Binary signal monitor (input MUX)	128 signals monitored
Binary signal control (output MUX)	64 signals controlled
System state correlator	8 signals correlated
Trigger delay counter	0 to 55 μs, simulated
High-speed buffer	2K (16 bits) words monitored (0.5 μs)
Low-speed buffer	128 signals monitored serially (50 μs)
Sampling resolution	50 μs--input high-speed MUX words (16 bits)
	350 μs--input low-speed MUX words (112 bits)
	1000 μs--output MUX words (64 bits)

the initial software package were provided by personnel of Harry Diamond Laboratories (HDL). In particular, an option was incorporated to provide the immediate on-line data printout using the TTY control unit in lieu of the line printer.

During these earlier portions of BDACS operations, the basic "background" software operating system was the Real Time Disk Operating System RDOS-REV 1.00 supplied by DGC. Following the Delta test program, an updated and improved operating system, RDOS-REV 3.01, was obtained from DGC and placed on BDACS. However, because the core resident portion of RDOS-REV 3.01 is substantially larger than that of RDOS-REV 1.00, the 16K core capacity of BDACS was insufficient to hold both RDOS-REV 3.01 and the BDACS control program concurrently. In fact, even with the earlier RDOS-REV 1.00, the concurrent loading of this operating system with the BDACS control program filled all but approximately 20 words of BDACS core. Consequently, there was no room for any substantial software additions or improvements to the BDACS control program as required for additional monitoring capability, without drastically reorganizing the control program.

Furthermore, the initial BDACS software package was not able to meet the design objective of a 50-μs minimum sample cycle time and was very inefficient both in operating time and in data storage requirements during the data reduction phase of the program.

For these reasons, a complete redesign of the control program package was initiated. The package described in the following sections has been implemented as the current BDACS control program. The control program is written in the DGC-supplied RDOS Assembly language and interfaces with the RDOS-REV 3.01 operating system.

2. BDACS CONTROL PROGRAM--GENERAL ARCHITECTURE

The BDACS control program has been specifically designed in a "structured program" modular format that allows for future expansion or modification without complicated interactions or interdependencies among the different modules. In particular, the overall operation of the control program has been separated into major sequential tasks, with each task programmed as a separate overlay module of the control-program save file. Information exchange among the different overlay modules has been kept to a minimum and is in one of the following forms:

(a) ZREL parameters (a very limited number)

(b) NREL tables in the root binary

(c) data files organized under the RDOS operating systems

Moreover, those system parameters defined within each overlay module which may be subject to change with system augmentation or expansion have been collected into a separate system parameter file, BDACS.SR, organized under RDOS. As described below, this file should be assembled with each program segment when the user constructs the control-program save file.

The BDACS control program is composed of a root binary section and six overlay modules which are common to a single overlay region. The major tasks associated with the root binary and each overlay module are summarized below and are described in detail in the subsequent sections.

Module	Name	Task
Root binary	MONTR.SR	holds ZREL parameters
		holds NREL tables
		provides overlay control
Overlay No. 1	PHAS1.SR	initializes and opens RDOS files
		reads and constructs METHOD file data acquisition and reduction tables

Module	Name	Task
		reads and constructs METHOD file data control tables
		reads and stores PREAMBLE query file
		stores the METHOD and ASSIGN files
Overlay No. 2	PHAS2.SR	controls real-time data acquisition and control sequencing
Overlay No. 3	PHAS3.SR	retrieves and stores HSB data
Overlay No. 4	PHAS4.SR	reduces and stores the MUX and LSB data
Overlay No. 5	PHAS5.SR	reads and stores POSTSCRIPT query file
		produces the on-line data printout
Overlay No. 6	ERMSG.SR	generates error messages
		provides normal or abnormal return to the RDOS-CLI

To produce the working load module (that is, the .SV save file and the .0L overlay file) of the control program, the user should first assemble the root binary and each overlay module using the RDOS-supplied assembler to produce the associated relocatable binary (.RB) file. The RDOS-CLI command strings for these assemblies are

(a) Root binary

```
ASM BDACS MONTR MONTR.RB/B
```

(b) Overlay No. 1

```
ASM BDACS PHAS1 PHAS1.RB/B
```

(c) Overlay No. 2

```
ASM BDACS PHAS2 PHAS2.RB/B
```

(d) Overlay No. 3

```
ASM BDACS PHAS3 PHAS3.RB/B
```

(e) Overlay No. 4

```
ASM BDACS PHAS4 PHAS4.RB/B
```

(f) Overlay No. 5

```
ASM BDACS PHAS5 PHAS5.RB/B
```

(g) Overlay No. 6

```
ASM BDACS ERMSG ERMSG.RB/B
```

Following assembly, the relocatable binary files are combined, and relocatable externals are resolved to produce the final save and overlay files. The RDOS-CLI command string for this loading operation is

```
RLDR MONTR [PHAS1,PHAS2,PHAS3,PHAS4,PHAS5,ERMSG]10/ C .
```

The final result is the generation of a save file named MONTR.SV and an associated overlay file named MONTR.0L.

3. BDACS DATA FILES AND TEMPORARY FILES

The BDACS control program uses the RDOS file management routines to create and maintain a number of different data files and temporary files used during BDACS operations. A general description of the content and format of these files follows.

PREAMBLE.DA--An RDOS sequentially organized, permanent, and write-protected file containing alphanumeric text information generated and maintained by the DGC-supplied text editor (EDIT.SV) under the control of the BDACS operator. The file contains a series of alphanumeric "query" or "statement" lines which are presented to the BDACS operator line by line at the beginning of a BDACS test run. A query line requires operator response, and both the query and response are stored for later incorporation in the final data vector. A statement line (which begins with an asterick) does not require a response. It is used to incorporate information into the final data vector which does not change from one test run to the next or it is used for formatting purposes.

POSTSCRIPT.DA--An RDOS sequentially organized, permanent, and write-protected file, similar to the preamble file described above. The postscript file contains a series of query or statement lines which are presented to the BDACS operator following a BDACS test run. Again, the query/response or the statement is incorporated in the final data vector.

ASSIGNA.DA--An RDOS randomly organized, permanent, and write-protected file containing alphanumeric text information generated and maintained by the ASSIGN.SV program which is supplied as part of the BDACS software package. The alphanumeric text is used to describe the signals being monitored by the HSB. In particular, the textual information for each signal is contained in 16 bytes (8 words packed mode 1) null filled. Thus, information for 32 signals is held in a 256-word disk block, and the file contains as many disk blocks as required for the HSB signals.

ASSIGNB.DA--An RDOS randomly organized, permanent, and write-protected file containing alphanumeric text information similar to that described above. For the present file, the text is used to describe the signals being monitored by the BDACS MUX panels. Again, each signal text is packed in 8-word segments, with 32 signal texts per 256-word disk block.

ASSIGNC.DA--An RDOS randomly organized, permanent, and write-protected file containing alphanumeric text information similar to that described above. For this file, the text is used to describe the signals being monitored by the BDACS LSB. Again, each signal text is packed in eight-word segments, with 32 signal texts per 256-word disk block.

BDACS.DA--An RDOS contiguously organized, permanent, write- and attribute-protected file containing the binary data acquired during a BDACS test run. The binary information is written into the file space independent of and external to the RDOS operating system and file maintenance routines. However, during the data reduction phase of the test run, the data are retrieved from the file under control of RDOS routines.

TEMPA.TM--An RDOS sequentially organized temporary file, created by the BDACS control program at the beginning of each BDACS test run. The file provides temporary storage for the query/response lines and the statement lines of the preamble file described earlier. At the end of a BDACS test run, this file is transferred to the final data vector and/or the on-line data printout.

TEMPB.TM--An RDOS randomly organized temporary file, created by the BDACS control program preceding the retrieval of the HSB data. At the end of a BDACS test run, this file is transferred to the final data vector and/or the on-line data printout.

TEMPC.TM--An RDOS randomly organized temporary file, created by the BDACS control program preceding the reduction of the MUX and LSB data (provided the magnetic-tape option is not in effect). At the end of a BDACS test run, this file is transferred to the on-line data printout.

TEMPD.TM--An RDOS sequentially organized temporary file, created by the BDACS control program following data reduction. The file provides temporary storage for the query/response lines and the statement lines of the postscript file described earlier. At the end of a BDACS test run, this file is transferred to the final data vector and/or the on-line data printout.

4. BDACS METHOD FILES

The BDACS operator provides information to the BDACS control program on how a test run is to be conducted (or controlled) through a "method" file. The different method files are RDOS sequentially organized files containing alphanumeric text information generated and maintained by the DGC-supplied text editor (EDIT.SV) under the control of the BDACS operator. Each method file should contain the lines of information that follow.

(1) The identification line does not provide information to the control program but serves to identify the contents of the file.

(2) The sample rate line should contain a single-precision, decimal integer between 50 and 4095 which represents the sample rate (in microseconds) for the BDACS MUX panels. The integer may be preceded and followed by alphabetic text strings (nonnumeric).

(3) The test duration line should contain a double-precision, decimal integer which represents the total duration of the test run (in milliseconds). The integer may be preceded and followed by alphabetic text strings (nonnumeric).

(4) The HSB line should contain a single-precision, decimal integer between 500 and 10000 which represents the sample rate (in nanoseconds) for the BDACS HSB. An asterisk immediately following the integer indicates that the HSB data should be transferred to the on-line data printout file following the data run. Alternatively, an integer of zero in this line indicates that the HSB data-retrieval phase will be bypassed. The integer may be preceded and followed by alphabetic text strings (nonnumeric).

(5) The LSB line should contain a single-precision, decimal integer between 1 and 16 which represents the MUX input number (on the high-speed word) at which the LSB signal is being serially monitored. Alternatively, an integer of zero in this line indicates that the LSB is not in use. The integer may be preceded and followed by alphabetic text strings (nonnumeric).

(6) The MUX reported points list consists of two or more lines of information representing those MUX input points which are selected for data reduction and/or inclusion in the on-line data printout following a test run. The list begins with a "header" line that is used for formatting but imparts no information to the control program. Following the header line is a series of data lines that contain one or more single-precision, decimal integers separated by non-digit characters. The integers, ranging from 1 to 128, represent those points on the input MUX panels to be included in the data-reduction phase of the control program. An asterisk immediately following an integer is a flag indicating that the specified point should also be contained in the on-line data printout. The last line of the MUX reported points list should be an alphabetic text string (for example, END OF MUX LIST.), containing no digits, which is used as an end-of-list indicator.

(7) The LSB reported points list like that described above, consists of two or more lines of information representing those LSB input points that are selected for data reduction and/or inclusion in the on-line data printout. Again, the list consists of a header line, a series of data lines, and an end-of-list indicator line. The integers in the data lines represent those LSB input points to be included in the data-reduction phase of the control program. An asterisk immediately following an integer is a flag indicating that the specified point should be included in the on-line data printout.

(8) The control point list consists of one or more lines of information representing the control functions to be performed by the control program during a test run. The list begins with a header line that is used for format but imparts no information to the control program. Following the header line is a series of data lines, each of which contains two single-precision, decimal integers followed by a double-precision, decimal integer. The first single-precision integer represents the state which a point on the MUX output panel is to assume, either 0 or 1. The second single-precision integer represents the point number on the MUX output panel, ranging from 257 to 318. However, a special point number of 0 may also be used. This number does not correspond to any point on the MUX output panel; instead, it indicates when the control program should begin data storage during data acquisition. The double-precision integer represents the elapsed time (in milliseconds) from the beginning of a test run at which the selected control point is to assume the indicated state. The series of data lines should be arranged in an ascending sequence with respect to time, and no two adjacent times should differ by more than 32,767 ms. Furthermore, the last time in the list and the test duration time specified earlier should not differ by more than 32,767 ms. If necessary, pseudo-control data lines, which will not affect the current state of a control point, may be placed in the data series to comply with the above limitation. As a second limitation, the total number of lines in the data list should not be greater than 128. Last, an

end-of-list alphabetic text string may optionally be included in the list and as a final line of the method file. An example of a typical method file is shown in figure 2.

```
IDENTIFICATION: TEST METHOD
SAMPLE RATE (IN USEC):  100
DURATION (IN MSEC):     10000
HS BUFFER:      5500*
LS BUFFER:      16
REPORTED POINTS TABLE
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*
17*18*19*20*21*22*23*24*25*26*27*28*29*30*31*32*
33*34*35*36*37*38*39*40*41*42*43*44*45*46*47*48*
49*50*51*52*53*54*55*56*57*58*59*60*61*62*63*64*
65*66*67*68*69*70*71*72*73*74*75*76*77*78*79*80*
81*82*83*84*85*86*87*88*89*90*91*92*93*94*95*96*
97*98*99*100*101*102*103*104*105*106*107*108*109*110*111*112*
113*114*115*116*117*118*119*120*121*122*123*124*125*126*127*128*
END OF TABLE
LS BUFFER TABLE
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*
17*18*19*20*21*22*23*24*25*26*27*28*29*30*31*32*
33*34*35*36*37*38*39*40*41*42*43*44*45*46*47*48*
49*50*51*52*53*54*55*56*57*58*59*60*61*62*63*64*
65*66*67*68*69*70*71*72*73*74*75*76*77*78*79*80*
81*82*83*84*85*86*87*88*89*90*91*92*93*94*95*96*
97*98*99*100*101*102*103*104*105*106*107*108*109*110*111*112*
113*114*115*116*117*118*119*120*121*122*123*124*125*126*127*128*
END OF TABLE
STATE   SIGNAL  TIME
  1     0       0
  1     257     0
  1     258     0
  1     259     0
  1     265     0
  1     266     0
  1     267     0
```

Figure 2. Typical BDACS method file (partial).

5. BDACS DATA VECTOR AND ON-LINE PRINTOUT

The results of a BDACS test run may be incorporated in a final data vector written on magnetic tape and/or an immediate on-line data printout produced by the BDACS line-printer or TTY control unit.

5.1 BDACS Data Vector

If the "M" global switch option is in effect, the final data vector on magnetic-tape unit MTO contains eight files described below.

(a) MTO:0 is the preamble file query/response lines and statement lines transferred line by line from the TEMPA.TM file. This file has the standard RDOS tape format and record length of 514 bytes.

(b) MTO:1 is the method file selected for the particular test run which is transferred line by line. Again, the file has the standard RDOS tape format and record length of 514 bytes.

(c) MTO:2 is the ASSIGNA.DA file transferred as 256-word blocks. The record length of each block is 512 bytes.

(d) MTO:3 is the ASSIGNB.DA file transferred as 256-word blocks. The record length of each block is 512 bytes.

(e) MTO:4 is the ASSIGNC.DA file transferred as 256-word blocks. The record length of each block is 512 bytes.

(f) MTO:5 is the HSB data transferred as 256-word blocks from TEMPB.TM. The record length of each block is 512 bytes.

(g) MTO:6 is the MUX input panel data and LSB data generated during data reduction. The file consists of 768-word data blocks with record lengths of 1536 bytes. Each of the 256 data entries in the data blocks has six bytes (or three words). The first byte contains a state bit (either 1 or 0), an error-recovery bit (either 1 or 0) and a select bit (either 1 or 0). The second byte contains the MUX or LSB point number corresponding to the state. The third, fourth, fifth, and sixth bytes contain a double-precision integer representing the time from the beginning of the test run (in ms) at which the selected point changed to the prescribed state.

(h) MTO:7 is the postscript file query/response lines and statement lines transferred line by line from TEMPD.TM file. This file has the standard RDOS tape format and record length of 514 bytes.

If the "L" global switch option is in effect, the final data vector is transferred to the BDACS line printer. Similarly, if the "T" global switch option is in effect, the final data vector is transferred to the BDACS TTY control unit. For either case, this data vector contains the following entries:

(a) The preamble file query/response lines and statement lines transferred line by line from TEMPA.TM file.

(b) The postscript file query/response lines and statement lines transferred line by line from TEMPD.TM file.

(c) The method file transferred line by line from the method file selected for the test run.

(d) The HSB data, if the printout option as described in section 4 is in effect. This printout consists of (1) the assignment table for the different HSB input signals derived from ASSIGNA.DA, and (2) a chronological list of the monitored states of the HSB input signals. The time at which each signal was monitored is derived from the HSB sample rate described in section 4.

(e) The input MUX data and LSB data for those MUX and LSB signal points which were flagged for on-line data printout, as described in section 4. The printout consists of data lines containing (1) the new state (either 1 or 0) which a signal point assumes, (2) the time (in microseconds) at which the state change occurred, (3) the signal point number and select code (either "M" for MUX panel or "L" for LSB), and (4) the signal point mnemonic name for the signal as derived from either the ASSIGNB.DA or the ASSIGNC.DA files.

6. ROOT BINARY SEGMENT--MONTR.SR

The root binary segment is the part of the control program that remains core resident throughout a BDACS test run. For this reason, the root binary segment was designed to require only a small amount of core, while most of the program core requirements are shared among the several overlay modules. The assembly language structure for MONTR.SR is given in appendix A, section A-1. The root binary segment is divided into the following three regions (sect. 6.1, 6.2, and 6.3).

6.1 ZREL Parameters

This region is organized in relocatable page-zero locations and contains those parameters that are common among or global to the several overlay modules. In particular, the following parameters are presently in use.

(a) OVRTN holds the return address to the root binary from all overlay modules (except when program exit is affected in ERMSG.SR). This parameter is set in subroutine OVLOD as part of the overlay loading procedure.

(b) RECOV provides several functions: First, it holds the system error code if an RDOS system call error occurs during processing in any of the overlay modules. Second, it holds the root binary return address, previously stored in OVRTN, if a recoverable error occurs in any overlay module.

(c) ERRTN contains the NREL address to the root binary section labelled ERROR. This allows an abnormal exit from any overlay module to the root binary segment when specific errors occur.

(d) ERCOD contains an error code which is returned to the root binary segment whenever an abnormal error return is affected. In particular, a code of -1 is returned for an RDOS system call error. For other errors, the left-hand byte of ERCOD contains the overlay number in which the error condition occurred, and the right-hand byte of ERCOD contains the error number for that particular overlay module. ERCOD is set within each overlay module and supplies information to the error message overlay ERMSG.SR.

(e) MAGFG provides two functions: First, it holds the first global switch word supplied from the COM.CM file generated by the RDOS-CLI. Second, it holds a nonzero value as a flag if global switch "M" is optionally set indicating a magnetic tape is required. This parameter is set in overlay module PHAS1.SR and used in PHAS1.SR, PHAS3.SR, PHAS4.SR, and PHAS5.SR.

(f) PRINT provides two functions: First, it holds the second global switch word supplied from the COM.CM file generated by the RDOS-CLI. Second, it holds nonzero print flags if global switch "L" is set, for line printer required, or global switch "T" is set, for TTY control unit required. Bit 1B15 is set for "L" and 1B0 is set for "T." This parameter is set in PHAS1.SR and used in PHAS5.SR.

(g) SMPRT holds the MUX panel sample rate as a single-precision binary integer. It is set in overlay module PHAS1.SR and used in PHAS2.SR and PHAS4.SR.

(h) HSMON holds the HSB sample rate as a single-precision binary integer and the optional HSB on-line data printout flag in 1B0. It is set in overlay module PHAS1.SR and used in PHAS3.SR and PHAS5.SR.

(i) LSMON holds the LSB signal point on the MUX panel as a single-precision binary integer. It is set in overlay module PHAS1.SR and used in PHAS4.SR during data reduction.

(j) BLKCT contains the number of remaining track surfaces on the disk in which data from the MUX input panel are stored. The parameter is initially set to a maximum count, currently 200, and decremented during overlay module PHAS2.SR operations. The parameter is also used in PHAS4.SR during data reduction.

(k) HLDCT contains the number of data blocks acquired but not stored on the disk preceding the onset of data storage. The parameter is initially zero and periodically incremented during data acquisition PHAS2.SR of the control program. The parameter is used in PHAS4.SR during data reduction.

6.2 NREL Input/Output Control Tables

This region of the root binary segment consists of three tables containing input and output control information.

(a) MSKTB is divided into three parts. The first part contains mask words of those MUX input points which have been designated for data reduction (see sect. 4). The mask words are packed with 16 points per word, each bit from 1B15 to 1B0 being set if the corresponding point is reported. At present, eight such mask words corresponding to 128 input points are in use. The second part contains mask words whose bits correspond to the previous state (either 1 or 0) of the different MUX input points. As above, the words are packed with 16 points per word for a total of eight mask words. Initially, the mask words are all zero, corresponding to an assumed initial 0 or "off" state for the input points. The third part of MSKTB contains mask words of those MUX input points which have been designated for immediate on-line data printout (see sect. 4). Again, the words are packed with 16 points per word for a total of eight mask words. Thus, the total present length for MSKTB is 24 words, plus one spare.

(b) LSBTB is the LSB equivalent to MSKTB. Again the table is divided into three parts containing mask words which correspond to the reported points, the previous state, and the points enabled for on-line printout.

(c) CTLTB contains the information required to perform the control operations in overlay module PHAS2.SR. Currently, the table has a capacity for 128 such control functions, with each control function having a corresponding three-word entry in the table. The first word of an entry contains the elapsed time (in milliseconds) from the previous entry at which the present control function is to be activated. The second word of an entry contains a word mask specifying the corresponding bit position in the MUX output-panel buffer word which is to be activated (or altered). The third word of an entry contains both the state to be activated (in bit position 1B15) and the address of the MUX output-panel buffer word (in the remaining bit positions).

6.3 NREL Code

This region of the root binary segment contains the limited amount of instruction code which is required for overlay module initiation and loading and for overlay module entry and exit control.

The control program begins at parameter START and immediately initializes and opens the control program overlay file MONTR.OL on channel No. 7. Next, the overlay modules PHAS1.SR, PHAS2.SR, PHAS3.SR, PHAS4.SR, and PHAS5.SR are sequentially loaded into memory via the overlay load subroutine OVLOD. Following each load a branch instruction is performed into the overlay region to addresses OVST1, OVST2, OVST3, OVST4 and OVST5. Following a normal completion of these overlay module instructions, or if an error condition occurs, the last overlay module ERMSG is loaded and entered. From this overlay module a normal or abnormal return to the RDOS-CLI is performed, or the root binary segment is reentered if a recoverable error is encountered.

7. OVERLAY MODULE--PHAS1.SR

This overlay module is the part of the control program that receives the control input data from the "method" file and constructs the control tables as described in sections 4 and 6.2. In addition, this module performs the query/response operations associated with the PREAMBLE.DA file and begins construction of the BDACS temporary files and the BDACS data vector described in sections 3 and 5.

PHASE1.SR is composed of instructions interspersed with tables and parameters. Logically, the module is divided into three main regions, the "backbone," the subroutines, and the name/test strings. These regions are described in detail in sections 7.1 to 7.4 and the assembly language listing for PHAS1.SR is shown in appendix A, section A-2.

7.1 PHAS1.SR Backbone Code

The backbone code for PHAS1.SR is primarily a set of sequential initialization tasks for the control program. The main tasks performed are to

(a) reset MUX output buffer, MUX input mask table, and the LSB input mask table.

(b) disable RDOS spooling on $LPT and $TTO; open the output message unit ($TTO).

(c) open the RDOS-generated COM.CM file and retrieve the global switches and "method" file name; set data vector and on-line printout option flags (see sect. 6.1(f) and (h)); initialize MTO (if required).

(d) open the prescribed "method" file and extract program control data as described in section 4; construct MUX input mask table, LSB input mask table, and MUX output control table, as described in section 6.2.

(e) create TEMPA.TM file; open TEMPA.TM and PREAMBLE.DA files; read preamble query/statement lines and obtain operator response; construct TEMPA.TM.

(f) transfer TEMPA.TM, the "method" file, ASSIGNA.DA, ASSIGNB.DA, and ASSIGNC.DA to the magnetic-tape data vector (if "M" option is in effect).

(g) check for proper sense-switch conditions.

During the process of receiving initialization parameters and data, these quantities are checked for proper value limits. If an out-of-bounds condition arises, control is transferred to an error routine.

7.2 PHAS1.SR Subroutine Code

The subroutines which comprise part of the PHAS1.SR code are accessed from the backbone code one or more times using the assembly language JSR instruction. Typically, access to the subroutine is indirect, with the subroutine starting address stored in an access word near the JSR instruction. Moreover, depending on the subroutine, one or more parameters may follow the JSR instruction. Return is made to the instruction following the last parameter, unless errors or exceptional conditions cause branching to other parts of the code. The following paragraphs briefly describe the subroutines found in PHAS1.SR.

(a) TABLE reads the contents of the MUX or LSB-reported points list in the "method" file and constructs the MSKTB or LSBTB mask table described in section 6.2(a) and (b). Three parameters are associated with the subroutine: the first contains the beginning address of the mask table, the second contains the maximum size of a valid list entry, and the third contains offset within the table for the beginning of the on-line printout masks. Entries within the list are assumed to begin at a value of one. An exceptional return to the error routine occurs if an entry in the list is out of bounds. Normal return occurs when a line containing no digits is encountered in the list.

(b) CKNBR checks a binary number for valid bounds. The number is contained in AC0. The lower bound is given as the first parameter and the upper bound is given as the second parameter. An exceptional return to the error routine occurs if an out-of-bounds condition occurs.

(c) XFERF transfers ASCII data from one file to another. The two parameters that are required contain the byte pointers to the source file name and the destination file name. The files are opened on RDOS channels No. 3 and No. 4, the transfer is made on a line-by-line basis, and the files are then closed on both channels. No exceptional condition is explicitly contained in this subroutine.

(d) XBLK transfers contents from a disk file to a magnetic-tape file block by block (256 words per block). Similar to XFERF, the files in XBLK are opened on RDOS channels No. 3 and No. 4, the transfer is made, and the files are then closed on both channels.

(e) MVWUT transfers words to a destination storage area from the utility buffer pointed to by the byte pointer UTBPT. Two parameters are required, the first containing the address of the destination area, and the second containing the number of words to be moved.

(f) FMMSK forms a mask (in AC1) and displacement value (in AC0) corresponding to an initial binary value in AC0. In effect, the subroutine divides AC0 by 16 to form the displacement as the quotient. The remainder is used to set the associated bit position in the mask word contained in AC1 (a remainder of 0 sets bit 1B15). All other bits in the mask are reset.

(g) GTSPN reads an ASCII decimal integer text string and forms the corresponding single-precision unsigned binary integer in AC0. The byte pointer to the beginning of the text string is stored in AC1. The subroutine searches the text string until the first decimal character is encountered. Thereafter, each succeeding decimal character is folded into the binary equivalent of the number. The first non-decimal character terminates the scan, and upon exit from the subroutine AC2 contains the byte pointer to this terminating character. A jump is made to the error routine whenever a binary number overflow occurs (that is, a number greater than 65,383) or whenever no number is found within the text string.

(h) GTDPN is similar to GTSPN except that a double-precision unsigned binary integer is formed in AC0 and AC1. An overflow occurs whenever the number is greater than 4,294,967,295.

(i) GTSPR is similar to GTSPN except that if no number is found in the text string, then exit from the subroutine is performed via the first parameter.

(j) GTBYT retrieves a byte from a text string pointed to by a byte pointer in AC2. The byte is returned in AC0 (right-hand position), and the byte pointer is left unchanged.

(k) STBYT stores a byte held in AC0 (right-hand position) in the text string pointed to by a byte pointer in AC2. Upon return from the subroutine, the byte pointer is incremented by one count.

(l) CREAT creates a sequentially organized disk file by using the system call .CREAT. The first parameter contains a name pointer to the name of the file to be created. If the file already exists, the old file is deleted and a new file is created; otherwise, all system errors cause a branch to the system error routine.

(m) DELET deletes a file by using the system call .DELET. The first parameter contains a name pointer to the name of the file to be deleted. If the file does not exist, no error is initiated; otherwise, all system errors cause a branch to the system error routine.

(n) OPFLE opens a file on an RDOS channel by using the system call .OPEN. The first parameter contains a name pointer to the name of the file to be opened. The second parameter contains the channel number. Default characteristics for the file are assumed when the subroutine is called. All system errors cause a branch to the system error routine.

(o) CLFLE closes a file on an RDOS channel by using the system call .CLOSE. The first parameter contains the channel number to be closed. All system errors cause a branch to the system error routine.

(p) INMTA initializes the magnetic tape until MTO by using the system call .INIT. A partial initialization is assumed when the subroutine is called, and the subroutine waits locally until the "device ready" status is achieved. All system errors cause a branch to the system error routine.

(q) WTLUT writes a line from the utility buffer (pointed to by byte pointer UTBPT) to the file opened on channel No. 3. This subroutine uses the system call .WRL. All system errors cause a branch to the system error routine.

(r) TYLUT is similar to WTLUT except the channel on which the system console output file ($TTO) is opened is No. 1.

(s) TYPMG is similar to WTLUT and TYLUT except the byte pointer is given as the first parameter and points to a message text string.

(t) RDSUT reads sequentially a number of bytes from the file opened on channel No. 4 to the utility buffer (pointed to by byte pointer UTBPT). The subroutine uses the system call .RDS, and the number of bytes to be read is given as the first parameter. All system errors cause a branch to the system error routine.

(u) RDLUT reads a line from the file opened on channel No. 4 to the utility buffer (pointed to by byte pointer UTBPT). The subroutine utilizes the system call .RDL, and upon successful return, AC2 contains the byte pointer to the byte following the end of the line. All system errors cause a branch to the system error routine.

(v) RDLUR is similar to RDLUT except that, upon detecting an end-of-file error condition, the subroutine exits via the first parameter.

(w) RDLUA is similar to RDLUT except that the byte pointer is located in AC2 and the channel number for the file is given as the first parameter.

(x) SPLDS disables spooling on a device whose name is pointed to by the byte pointer given in the first parameter. The subroutine uses the system call .SPDA, and all system errors cause a branch to the system error routine.

(y) OPMTA opens a magnetic tape file for free format I/O on an RDOS channel by using the system call .MTOPD. The first parameter contains a byte pointer to the name of the magnetic-tape file name, and the second parameter contains the channel number on which the file is to be opened. All system errors cause a branch to the system error routine.

(z) MTAWT performs a magnetic-tape free format "write" operation to the file on channel No. 3 by using the system call .MTDIO. The data to be transferred are located in the buffer pointed to by BUFFR and include a block of 256 words. All system error conditions except "end-of-file" cause a branch to the system error routine.

(aa) MTASR is similar to MTAWT, except a free format "space reverse" of one record is initiated.

(bb) MTAEF is similar to MTAWT, except a free format "write end-of-file" is initiated.

(cc) RDBLK reads a disk block from the file opened on channel No. 4 using the system call .RDB. The first parameter contains the block number within the file which is to be read, and the second parameter contains a branching address for an end-of-file error condition. A single 256-word block is read from the file into the core area pointed to by BUFFR. All system errors except the end-of-file condition cause a branch to the system error routine.

7.3 PHAS1.SR Name/Text Strings and Buffers

The final region of PHAS1. SR is composed of a set of packed ASCII name and text strings pointed to by name or text byte pointers. These pointers are assembly language "equates" that do not generate an object code, but instead provide byte pointer values used in the "backbone" and subroutine regions of PHAS1.SR. The name and text strings are those RDOS file names and BDACS file names or messages used in PHAS1.SR and are generated with assembly packing mode 1, the normal packing mode for RDOS strings. Last, the final part of the name and text string region contains the address pointer UTBPT to a 68-word utility buffer area and the address pointer BUFFR to a 256-word block I/O buffer. These buffers are used for I/O operations for several subroutines described in section 7.2.

7.4 PHAS1.SR Error Routine

Several program error conditions can cause a branch to the PHAS1.SR error routine. This routine determines the point in the PHAS1.SR code where the error occurred and encodes an error code word which is stored in ERCOD within the root binary segment (see sect. 6.1(d)). The error code word contains in the left-hand byte the overlay module number (one in the present case) in which the error occurred and in the right-hand byte the displacement from the beginning of an error table. This error table (pointed to by the symbol ERTBL) contains as entries those addresses which immediately follow a JSR instruction to the error routine. Within the routine, the error table is scanned until a match is found between an error table entry and AC3, the latter containing the return address of that JSR instruction from which the routine was entered. Thus, the error is identified as a displacement from the start of the error table. A corresponding displacement from the start of a message pointer table in the overlay module EMMSG.SR contains a table entry which points to the associated error message. This message is subsequently printed on the system output console (that is, $TTO).

At present, the 12 program errors detected for PHAS1.SR are

(a) ERR1--"MONITOR POINT UNDERFLOW"--the entry in the MUX input list or LSB input list is less than unity.

(b) ERR2--"MONITOR POINT OVERFLOW"--the entry in the MUX input list or LSB input list is greater than maximum value allowed (presently 128 and 128, respectively.

(c) ERR3--"CONTROL TABLE OVERFLOW"--the number of control table entries exceeded the maximum value allowed (presently 128).

(d) ERR4--"OUTPUT LIST NOT IN SEQUENTIAL ORDER"--the entries in the control table are not monotonically increasing in time.

(e) ERR5--"OUTPUT LIST DELTA TIME OVERFLOW"--the time difference between two adjacent control list entries is greater than the maximum value allowed (32,767).

(f) ERR6--"CONTROL POINT UNDERFLOW"--the MUX output point in a control table entry is less than the minimum value allowed (presently 257).

(g) ERR7--"CONTROL POINT OVERFLOW"--the MUX output point in a control table entry is greater than the maximum value allowed (presently 318).

(h) ERR8--"SP-DP OVERFLOW"--the number in a text string exceeded the single precision maximum (65,535) or the double precision maximum (4,294,967,295) in subroutines GTSPN (or GTSPR, GTSPA) and GTDPN, respectively.

(i) ERR9--"EOL, NO NUMBER FOUND"--no number was found in a text string when subroutines GTSPN, GTSPA, or GTDPN are called.

(j) ERR10--"SAMPLE RATE OUT OF BOUNDS"--the sample rate for the MUX panels is either less than the minimum allowed value (presently 50) or greater than the maximum allowed value (presently 4095).

(k) ERR11--"LOW SPEED BUFFER MONITOR POINT OUT OF BOUNDS"--the point on the input MUX panel where the low-speed buffer signal is monitored is less than the minimum value allowed (presently 1) or is greater than the maximum value allowed (presently 16).

(l) ERR12--"HIGH SPEED BUFFER SAMPLE RATE OUT OF BOUNDS"--the value specified for the HSB sample rate is less than the minimum value allowed (presently 500) or is greater than the maximum value allowed (presently 10,000).

If the error is a result of a system error returned from one of the RDOS system calls described in section 7.2, the error code ERCOD is set to -1 as a system error flag. For either type of error (program or system) return is made to the root binary section via ERRTN, as described in section 6.1(c).

8. OVERLAY MODULE--PHAS2.SR

This overlay module is the portion of the control program that initiates the start of data acquisition and supervises the operations of

the input and output MUX panels (multiplexers). This module has the complicated task of retrieving data from the input MUX and storing the data in a temporary buffer and, when the buffer is full, of transferring the data to the disk. Concurrently, the module is monitoring the real-time clock (RTC) and, at the preselected control times, updating the output MUX buffer data. These data are periodically sent to the output MUX panel to control the various BDACS operations during a test run.

Since the input and output data flow can take place at very fast rates (two input data words and one output data word can be transferred within 50 μs), the response of the standard RDOS interrupt service routine is not adequate. Thus, the main portion of the PHAS2.SR code contains a more streamlined interrupt service routine to handle the multiplexer, disk, and RTC operations. In addition, PHAS2.SR contains the test-run initiation and control loop code and data tables necessary to "set up" the multiplexer I/O controller. These three functions are described in more detail in the following sections, and the assembly language listing for PHAS2.SR is shown in appendix A, section A-3.

8.1 PHAS2.SR Multiplexer Controller Setup

The BDACS multiplexer controller contains the circuitry which provides the interface between the central processing unit (CPU) and the MUX input and output panels. Since this controller operates by direct memory access (DMA) data transfer, it must be preprogrammed with the appropriate core buffer address and word count for DMA. Furthermore, the information which controls the operation of MUX input and output panels must be supplied to the controller. These data are held in a PHAS2.SR data table and are transferred from the CPU to the multiplexer controller with the basic NOVA I/O instruction repertoire, as described below.

(a) DMUXA holds the input and output MUX panel starting addresses and word counts. A four-bit address and a four-bit word count are associated with each type of panel. The information is supplied to the controller by a DOA instruction.

(b) DMUXB holds the DMA input starting address for the initial DMA operation. The starting address points to a core buffer region in which the input data are stored. The address is supplied to the controller by a DOB instruction.

(c) DMUXC holds the highspeed word option bit (1B1) and the sampling rate for the controller (in bits 1B4-1B15). Also, for proper transfer of the data, 1B0 flag bit should be "set." The data are supplied to the controller by a DOC instruction.

(d) DMUXD contains the (negative) word count for the initial DMA operation. As noted above, this word count in conjunction with the starting address (DMUXB) controls the storing of input data in the core buffer. When the count is decremented to zero, a MUX interrupt occurs. In contrast to (c), a 1B0 flag bit should be "reset" for proper transfer of the data by a DOC instruction.

(e) DMUXE contains the (running) starting address for the remaining DMA operations. It typically contains a core address which is one less than that specified (in DMUXB). It is used to reset the running value of the address register in the controller following a MUX interrupt and overflow. Again, the data are supplied with a DOB instruction.

(f) DMUXF contains the (running) negative word count for the remaining DMA operations. It typically contains a count one greater (in absolute magnitude) than that specified in DMUXD. It is used to reset the running value of the word-count register in the controller following a MUX interrupt and overflow. Again, the data are supplied with a DOC instruction with 1B0 "reset."

Two I/O instruction special functions should also be supplied to the controller following the general reset of the controller. First, the START function may be supplied with any of the instructions given above and causes a reset of the DONE state and a set of the BUSY state in the controller. Second, the PULSE function is supplied to the controller immediately upon receipt of the BDACS operators signal to begin the test run (via the CPU console sense switch 1B0) and causes the controller to commence I/O operations.

8.2 PHAS2.SR Test-Run Initiation and Control Loop

The sequential tasks performed in PHAS2.SR are

(a) wait for all TTY and disk operations to end; reset the multiplexer controller, disk, and RTC,

(b) save the RDOS interrupt service routine pointer and enable the BDACS data acquisition interrupt service routine pointer and mask,

(c) position the disk at the beginning of the BDACS data storage file (see sect. 3.6),

(d) set up the multiplexer controller as described in section 8.1; set the RTC for 1 ms operation, and

(e) send a "ready" message to the BDACS operation on the TTY control unit; wait for the operator to initiate the test run; then start the RTC and MUX and branch to the control loop.

These tasks are performed within the control loop:

(a) Check the remaining time count of the current table entry (see sect. 6.2(b)); if the count is zero, update the appropriate MUX output buffer word or set the appropriate special flag (for example, HLDFG is set when the disk storage holdoff time is complete; ENDFG is set when the end of data-acquisition time is complete). Move to the next table entry.

(b) Check the RTC counter (RTCCT); if the count is nonzero, update the remaining time count for the current control table entry. Loop back to (a).

When data acquisition is ended, the data input buffer currently being filled is completed and transferred to the disk. The final tasks involve a general reset of the multiplexer controller, disk, and RTC and a replacement of the previously saved RDOS interrupt service routine.

8.3 PHAS2.SR Interrupt Service Routine

The special BDACS interrupt service routine for PHAS2.SR processes interrupts generated by either the multiplexer controller, the disk, or the RTC. This routine uses only accumulators AC0, AC1, and CARRY; thus, only these quantities need to be stored when entering the routine. The method by which each type of interrupt is handled is

(a) RTC interrupt is very simply handled by restarting the clock with the NIOS instruction and incrementing the RTC counter (RTCCT). Return is then made to the interrupted program.

(b) Disk interrupt first obtains the disk status word and checks for any error condition. The status word also determines which of two possible disk conditions initiated the interrupt. First, if a disk SEEK to a new track has just been completed, the service routine determines if a core buffer is full and the data in the buffer need to be transferred. If so, the transfer (disk WRITE) operation is initiated. The disk flag (DKPFG) is updated (for example, DKPFG=0 "idle," DKPFG=-1 "seeking," DKPFG=+1 "writes deferred," DKPFG=MUXFG "writing"), and return is made to the interrupted program. Second, if a disk WRITE operation has just been completed, the block count BLKCT (see sect. 6.1(j)) is decremented; the end of acquisition flag ENDFG is checked, and final closeout is made if ENDFG has been set.

The interrupt service routine then determines if a new disk track is required. If so, the disk SEEK is initiated, the disk flag described above is updated, and return is made to the interrupted program (or to the delayed WRITE initiating section, if required).

(c) MUX interrupt first obtains the MUX status word and restarts the controller. The status word is stored as the updated MUX flag (MUXFG) and compared to the previous status word for possible errors. Next, the disk storage holdoff flag (HLDFG) is checked. If reset, buffer transfer operations to the disk are inhibited and the holdoff counter (HLDCT, see sect. 6.1(k)) is incremented. If set, the disk flag (DKPFG, described above) is checked for "idle." If the disk is idle, the disk WRITE command is initiated, the disk flag is updated, and return is made to the interrupted program. If the disk is not idle, the disk flag is placed in "write deferred" and return is made to the interrupted program.

8.4 PHAS2.SR Error Routine

The error routine operation for PHAS2.SR is almost identical to that for PHAS1.SR (see sect. 7.4). In the present case, no RDOS system calls (or errors) occur. However, four program errors are monitored at present as described below.

(a) ERR1--"DISK ERROR"--1B15 of the disk status word was set.

(b) ERR2--"DISK OVERFLOW"--the maximum block transfer count was reached. Presently, this count is 2400 256-word disk blocks (that is, 100 complete tracks on the disk). This is a recoverable error in which present data acquisition stops, but BDACS control program processing of the data is initiated.

(c) ERR3--"MULTIPLEXER ERROR"--two consecutive multiplexer status words were identical.

(d) ERR4-"INPUT BUFFER OVERRUN"--two adjacent "deferred write" requests were attempted.

9. OVERLAY MODULE--PHAS3.SR

This overlay module contains the part of the BDACS control programs that supervises the retrieval and storage of the HSB data gathered during a test run. These data are presented to the high-speed word of the input multiplexer panel on an alternate set of 16 parallel data lines, after first being transmitted serially from the remote HSB storage unit to the HSB receiver unit adjacent to the multiplexer panels. An electronic switch controlled by point No. 319 of the

multiplexer output panel selects this alternate set of input data lines when the point on the output panel is activated or "set." Alternatively, the switch selects the normal set of input data lines when the point on the output panel is deactivated or "reset." Moreover, control of the serial transmission of each HSB data word is provided by point No. 320 of the multiplexer output panel. In particular, a command is initiated to transfer a new data word when the point on the output panel changes from "reset" to "set." Following this command, a delay of 4 ms is required to allow the data to be transferred and recombined at the HSB receiver circuits. Thus, PHAS3.SR must correctly control the states of both multiplexer output points No. 319 and 320.

PHAS3.SR is constructed along the same principles as PHAS2.SR described in section 8. In particular, a special-purpose interrupt service routine is used to control the multiplexer controller operations during the HSB data retrieval and storage. Following data retrieval, the standard RDOS interrupt service routine is reinstated. The stored HSB data are then transferred to the temporary disk data file TEMPB.TM described in section 3.8. Furthermore, if the final data vector is to be written on magnetic tape, the stored HSB data are also transferred to MTO:5, as described in section 5.1(f).

As with its predecessors, PHAS3.SR is composed of three parts, the main (backbone) code, the subroutines (including the interrupt service routine), and the text strings and buffers. The main features of these units are described below, and the assembly language listing for PHAS3.SR is shown in appendix A, section A-4.

9.1 PHAS3.SR Backbone Code

The PHAS3.SR backbone code includes both instructions and tables of data used during the execution of the instructions. Of particular importance are the parameters used to initiate and control the operation of the multiplexer controller.

(a) DMUXA = 000060 initiates the controller to accept one word from the MUX input panel at MUX address corresponding to the high-speed word. It is this word which has the alternate set of data lines to the HSB receiver. At the same time, the last word on the MUX output panel is activated. This word contains output control points No. 319 and 320 used to control HSB retrieval operations.

(b) DMUXB = INPUT-1 contains the beginning of a two-word input buffer where HSB data are initially deposited.

(c) DMUXC = 140144 allows the multiplexer to operate in the LS/HS mode at a 144_8-μs sample time.

(d) DMUXD = 077776 contains the (negative) number of input words (-2) received before a multiplexer controller interrupt occurs.

(e) DMUXE = INPUT-2 contains the running value of (b).

(f) DMUXF = 077775 contains the running value of (d).

(g) DMUXG = 147777 contains the alternate data to (c). For the present value, a 7777_8-μs sample time is sufficient for the serial data transmission from the HSB transmitter to the HSB receiver.

(h) CNTLL = 137777 contains the control word-pattern setting control point No. 319 "on" and point No. 320 "off."

(i) CNTLH = 037777 contains the alternate control word pattern to (h). The present value allows the setting of both point No. 319 "on" and point No. 320 "on."

(j) INPUT represents the address of the two-word input buffer area for DMA storage of the incoming HSB data.

The sequential tasks performed by the PHAS3.SR backbone code are summarized below.

(a) Reset the HSB output buffer pointed to by HSBUF and the MUX output buffer pointed to by MUXOB.

(b) Wait for all TTY and disk operations to end, save the RDOS interrupt service routine pointer and mask, and install the PHAS3.SR interrupt service routine pointer and mask.

(c) Initialize the multiplexer controller.

(d) Send an "HS Buffer Retrieval" message to the TTY console.

(e) Start the multiplexer controller, retrieve the HSB data, and store the data in the output buffer area pointed to by HSBUF. Continue operations until the buffer count (presently 2048 words) reaches zero.

(f) Transfer the buffer to disk file TEMPB.TM and, if the magnetic-tape option "M" is in effect, transfer the buffer to the tape file MT0:5.

9.2 PHAS3.SR Subroutine Code

Most of the subroutines encountered in PHAS3.SR have essentially the same structure as those subroutines described in section

7.2 for PHAS1.SR. In particular, the following subroutines are found in PHAS3.SR.

(a) CRRFL creates a randomly organized disk file by using the system call .CRAND. The first parameter contains a name pointer to the name of the file to be created. If the file already exists, the old file is deleted and a new file is created; otherwise, all system errors cause a branch to the system error routine.

(b) DELET deletes a file by using the system call .DELET. It is identical to subroutine DELET found in PHAS1.SR (see sect. 7.2(m)).

(c) OPFLE opens a file on an RDOS channel by using the system call .OPEN. It is identical to subroutine OPFLE found in PHAS1.SR (see sect. 7.2(n)).

(d) CLFLE closes a file on an RDOS channel by using the system call .CLOSE. It is identical to subroutine CLFLE found in PHAS1.SR (see sect. 7.2(o)).

(e) OPMTA opens a magnetic-tape file for free format I/O on an RDOS channel by using the system call .MTOPD. It is identical to OPMTA found in PHAS1.SR (see sect. 7.2(y)).

(f) MTAWT, MTASR, and MTAEF provide free format magnetic-tape operations using the system call .MTDIO. They are identical to MTAWT, MTASR, and MTAEF found in PHAS1.SR (see sect. 7.2(z), (aa), (bb)).

(g) RDBLK reads a disk block from a file opened on channel No. 4 using the system call .RDB. It is identical to the subroutine RDBLK found in PHAS1.SR (see sect. 7.2(cc)).

(h) TMPOT writes a series of disk blocks to a file opened on channel No. 3 using the system call WRB. The beginning of the first block corresponds to the beginning of the HSB output buffer pointed to by HSBUF. The number of blocks to be written is given by BLKCT (presently set to eight blocks for the HSB output buffer, corresponding to 2048 words in the buffer). All system errors cause a branch to the system error routine.

(i) INTSR contains the interrupt service routine for PHAS3.SR. In this routine, only the multiplexer controller is enabled for interrupts. When a multiplexer interrupt occurs, the controller is immediately restarted (BUSY is "set" and DONE is "cleared"), and the controller status is read and checked. Depending on the status word

flag 1B0, the data previously stored in the input buffer (INPUT) are valid or invalid. Thus, with 1B0 set, the data word is invalid and is not stored in the HSB output buffer. On the other hand, with 1B0 reset, the data word is valid and is stored. In either case, the alternate control data word is updated in the output MUX buffer, and the alternate sample time word is transmitted to the controller. A normal exit is made from the interrupt service routine, except after the storage of last data word of the HSB output buffer.

9.3 PHAS3.SR Error Routine

The error routine for PHAS3.SR is identical to that for PHAS1.SR (see sect. 7.4). For the present case, the error code word generated and stored in ERCOD has overlay module No. 3 coded in the left-hand byte. At present, only a single program error is detected by PHAS3.SR.

ERR1--"MULTIPLEXER ERROR"--two consecutive multiplexer words were identical.

10. OVERLAY MODULE--PHAS4.SR

This overlay module contains the part of the control program that performs a preliminary data reduction of the MUX and LSB data gathered and stored during the PHAS2.SR operation. These raw data were acquired (sampled) on a regular periodic basis as defined by the sampling rate specified by the BDACS operator (see sect. 4). As a consequence of this periodic sampling, the data signals may have been monitored many times in the same binary state, and much of the sampled data may be redundant. Therefore, the main task in PHAS4.SR is to scan the collected data, searching for changes in state for the monitored signals (i.e., changes from "0" to "1" or from "1" to "0"). When a state change is detected for a signal, the signal point number, the new state, and the time of the state change are all written into the BDACS data vector as a reduced data entry. This data reduction operation is applied only to those signals monitored with the MUX input panels and the LSB unit. Even then, only those signals which were previously specified by the BDACS operator are enabled for data reduction (see sect. 4). The HSB raw data are retrieved and stored in the data vector without any reduction, as described in section 9.

During data reduction, PHAS4.SR also supports an error detection and correction task. This task is performed by default unless the operator specifically disables error detection and correction by setting sense switch 1B1 on the CPU panel of the BDACS central processor unit. Error detection is achieved by a hardware strapping option which codes an address into each word of the input MUX panels. In particular, the

high-speed word of the input panel has 1B16 strapped "on" while the remaining low-speed words all have 1B16 strapped "off." Furthermore, bits 1B13 through 1B15 of the seven low-speed words are strapped with a binary number pattern from one to seven. This addressing scheme, although not the most efficient in preserving data positions on the input panels, retains the most data positions for the important high-speed word. When the data words are being reduced, the address of the word is checked for proper monitoring and storage sequence. If incorrect sequencing is detected, an error flag is set, and, if possible, a resequencing correction to the computed sample time is made. However, if the error appears too severe, data reduction is terminated, and a branch to the error routine is made.

As noted above, setting sense switch 1B1 disables error detection and correction. For this case, all bits of the data words are considered true data and not addresses. Data reduction is performed under the assumption that proper sequencing is maintained.

As with the previous overlay modules, PHAS4.SR may be logically separated into its backbone code, its subroutine code, and its text string and buffer area. Each of these regions is described more fully in the following paragraphs. The assembly language listing for PHAS4.SR is shown in appendix A, section A-5.

10.1 PHAS4.SR Backbone Code

The sequential tasks performed by PHAS4.SR are given in the following list.

(a) Clear the reduced data output buffer pointed to by the parameter MTABF.

(b) Open the raw data file BDACS.DA on channel No. 4. If the magnetic-tape "M" option is in effect, open magnetic-tape file MTO:6 on channel No. 3. Otherwise, create a random file TEMPC.TM, and open the file on channel No. 3.

(c) Determine the initial sequence time at which raw data were stored by using the data storage holdoff count HLDCT (see sect. 6.1(k)) and the sample rate SMPRT (see sect. 6.1(g)). Store the initial time in the buffer pointed to by TIMER.

(d) Determine the initial value of the previous low-speed word offset on the input MUX panels by using the data storage holdoff count HLDCT. The word offset is stored as PRADD.

(e) Read the sense switches. Branch to address L0 if 1B1 is "set" (i.e., if error detection and correction are suppressed); otherwise, branch to address LL1.

(f) Read and reduce the MUX and LSB raw data. Store reduced data entries in the buffer area pointed to by MTABF. When the buffer is full, transfer the buffer to the file opened on channel No. 3 (i.e., either MTO:6 if the magnetic tape is enabled, or TEMPC.TM temporary disk file). Continue until the number of data blocks specified by BLKCT (see sect. 6.1(j)) has been reduced.

(g) Close the files previously opened on channels No. 3 and 4, and return to the root segment program.

10.2 PHAS4.SR Subroutine Code

Many of the subroutines used in PHAS4.SR--especially those associated with file I/O which use RDOS system calls--are identical to those already described in sections 7 and 9 and will not be repeated here. Other subroutines unique to PHAS4.SR are described below.

(a) INTMR increments the present value of the sequence time (pointed to by TIMER). The incremental value is given by the sample rate SMPRT. Typically, the subroutine is called from the backbone code or from the subroutine RDBWD following the proper reduction of a high-speed word/low-speed word pair. This conforms to the actual sampling sequence which was performed in the data acquisition. Thus, the contents of TIMER contain the elapsed time from start of data acquisition to the time the datum (currently being reduced) was monitored. In addition, INTMR updates the low-speed buffer point number (LSPNT), the low-speed buffer word mask (LSWMK), and the low-speed buffer word offset (LSOFF). Again, in the actual acquisition, LSB data were serially transmitted with one data bit every sample cycle.

(b) RDBWD reduces the data word entered in accumulator AC0. Upon entry to the subroutine, accumulator AC2 contains the offset from the start of the MUX reported points mask table associated with the particular word. The mask table, located in the root binary segment, is pointed to by MSKTB and is described in section 6.2(a). The bit positions of the data word are first checked against the corresponding positions in the mask table to determine which points are enabled for data reduction. Those enabled points are then checked against the previous state values to determine if a state change occurred. Only if a state change occurs are the point number, the new state, and the current sequence time (given by the value of TIMER) transferred to the reduced data output buffer MTABF. Reduced data transferred to MTABF consist of three-word entries. The first word contains the point number, the state, the MUX/LSB flag, and the error recovery flag. The point number is in the RH position and the current state "0" or "1" is in bit position 1B0. If the datum is an LSB signal, bit position 1B1 is "1"; otherwise, 1B1 is "0" for an input MUX signal. If a sequence error was detected, bit position 1B7 is "1." The second and third words of

the output buffer entry contain the DP binary value of the current sequence time. The output buffer can hold 256 entries (768 words). When the buffer becomes full, the information is transferred to the file opened on channel No. 3 (either MTO:6 or TEMPC.TM), and the buffer is reset.

(c) LSBRD reduces data associated with the LSB. The subroutine is called from RDBWD whenever the LSB has been enabled for monitoring (LSMON is nonzero, see sect. 6.1(i)), and whenever the high-speed word of the input MUX panel is being reduced. The latter requirement is tested since the LSB data are received only on an input point of the high-speed word. Within the subroutine, the present LSB point number contained in LSPNT is checked against the associated bit position of the LSB mask table. The mask table, located in the root binary segment, is pointed to by LSBTB and is described in section 6.2(b). Those LSB point numbers which are enabled for data reduction are then checked against the previous state value. If a state change has occurred, the new state and the LSB point number (with 1B1 set to "1") is returned to the calling program (i.e., RDBWD).

(d) GTBWD controls the retrieval of the raw data from the BDACS.DA file using the system call .RDB. A complete data block of 3072 words is read to the data input buffer pointed to by BUFFR. The individual data words are then returned to the calling program in accumulator AC0 and the buffer pointer is updated each time the subroutine is called. When all data have been retrieved (as determined by the data block count BLKCT), a branch is made to address DONE to complete PHAS4.SR operations. As an added option within this subroutine, the sense switches on the CPU panel are read following the retrieval of the current data word. If sense switch 1B15 is "set," the raw data word is printed to the TTY console. Thus, the BDACS operator may selectively monitor the raw data which were acquired during a test run. This option may be useful for tracing possible error conditions.

(e) DMPWD generates the text string for the printout of the data word just described. The printout text consists of a set of 16 ASCII binary numbers corresponding to the 16 bit positions of the data word. The text string is stored in a buffer pointed to by byte pointer DPMSG.

10.3 PHAS4.SR Name/Text Strings and Buffers

The PHAS4.SR name and text strings are constructed in a manner similar to that described in section 7.3. The buffer areas used by PHAS4.SR are pointed to by the following parameters.

(a) MTABF is 768 words long and is used to hold the reduced data output entries (256 max) before they are transferred to the BDACS data vector.

(b) BUFFR is 3072 words long and is used to hold a raw data block previously written in the BDACS.DA file during data acquisition.

10.4 PHAS4.SR Error Routine

The error routine for PHAS4.SR operates identically to that for PHAS1.SR (see sect. 7.4). At present, four program errors are monitored as described below.

(a) ERR1--"CURRENT TIME OVERFLOW"--a double-precision overflow occurred for the number held in the current sequence time TIMER during initial timer setup.

(b) ERR2--"CURRENT TIME OVERFLOW"--a double-precision overflow occurred for the number held in the current sequence time TIMER during sequential updating.

(c) ERR3--"MAJOR SEQUENCE SLIP OCCURRED"--a noncorrectable sequence error was encountered. An indeterminate number of sequence step omissions occurred.

(d) ERR4--"MAX SEQUENCE ERROR COUNT EXCEEDED"--the maximum permissible number of sequence errors was exceeded (present errors set at 100).

11. OVERLAY MODULE--PHAS5.SR

This overlay module contains the part of the control program that performs the query/response operations associated with the POSTSCRIPT.DA file and completes construction of the BDACS data vector. In addition, PHAS5.SR provides the on-line data printout, if this printout is required.

As with the preceding overlay modules, PHAS5.SR is logically divided into three main regions: the "backbone" code, the subroutines, and the name/text strings and buffer area. These regions are described in detail in the following paragraphs, and the assembly language listing for PHAS5.SR is shown in appendix A, section A-6.

11.1 PHAS5.SR Backbone Code

The PHAS5.SR backbone code is primarily a set of sequential tasks designed to complete the BDACS data vector and generate the BDACS on-line printout. The main tasks are given in the following list.

(a) Open the console input file ($TTI) on channel No. 2; create the temporary file TEMPD.TM and open this file on channel No. 3.

(b) Open the POSTSCRIPT.DA file on channel No. 4; read the postscript query/statement lines and obtain the BDACS operator response; construct TEMPD.TM.

(c) Transfer TEMPD.TM to the magnetic-tape file MTO:7 (if the "m" option is in effect). Close the files on channels No. 2, 3, and 4.

(d) Open the on-line printout file on channel No. 2 ($TTO if "T" option is in effect, or $LPT if "L" option is in effect).

(e) Print the BDACS on-line "header" information; print the preamble and postscript information (obtained from TEMPA.TM and TEMPD.TM); print the "method" file which was employed for the current test run. See figures 3 and 4 for an example of these printouts.

(f) If the HSB unit was enabled, print the HSB assignment list (obtained from ASSIGNA.DA, see sect. 33). Print the HSB data obtained during the current test run (obtained from TEMPB.TM). See figure 5 for an example of these printouts.

(g) If either or both of the input MUX panels or the LSB unit contain signals which have been enabled for on-line printout, then open the assignment list files ASSIGNB.DA for the MUX signals and ASSIGNC.DA for the LSB signals on channels No. 4 and 5, respectively. Open the reduced data file on channel No. 3 (either MTO:6, if the "M" option is in effect, or TEMPC.TM).

(h) Read the entries in the reduced data file; construct a printout line consisting of the entry data and the signal mnemonic obtained from the appropriate assignment list file; continue processing the signals enabled for on-line printout until all data entries have been scanned. See figure 6 for an example of these printout lines.

(i) Close the files on channels No. 1, 2, 3, 4, and 5 and return to the root binary segment.

BDACS ONLINE PRINTOUT

PREAMBLE & POSTSCRIPT FILES

```
*       PREAMBLE FILE
*
 TEST IDENTIFICATION:  SAMPLE RUN
*SITE:  PICKENS, MISS.
 DATE:  12 JULY 76
 TIME:  1435
 BDACS STATUS:  OK
 ESS#1 STATUS:  OK
 TEMPS STATUS:  OK
 HSB USED?  Y
 LSB USED?  Y
 TRIGGER REQUIRED?  Y @ EMT*ZPT11*TP13
 TAPE OUTPUT REQUIRED?  Y
 ONLINE PRINTOUT REQUIRED?  Y
 ASSIGN FILES UPDATED?  Y
 COMMENTS:  NONE
*
*       END


*       POSTSCRIPT FILE
*
 TEMPS PULSE# 326
 TIME          1437
 BDACS STATUS:  OK
 ESS#1 STATUS:  UPSET
 TEMPS STATUS:  OK
 COMMENTS:  MEMORY ERROR; SWITCH OS FOR 2 MIN.
*
*       END
```

Figure 3. BDACS on-line printout of the PREAMBLE.DA and POSTSCRIPT.DA files (partial).

METHOD FILE

```
IDENTIFICATION: TEST METHOD
SAMPLE RATE (IN USEC):  100
DURATION (IN MSEC):     10000
HS BUFFER:      5500*
LS BUFFER:      16
REPORTED POINTS TABLE
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*
17*18*19*20*21*22*23*24*25*26*27*28*29*30*31*32*
33*34*35*36*37*38*39*40*41*42*43*44*45*46*47*48*
49*50*51*52*53*54*55*56*57*58*59*60*61*62*63*64*
65*66*67*68*69*70*71*72*73*74*75*76*77*78*79*80*
81*82*83*84*85*86*87*88*89*90*91*92*93*94*95*96*
97*98*99*100*101*102*103*104*105*106*107*108*109*110*111*112*
113*114*115*116*117*118*119*120*121*122*123*124*125*126*127*128*
END OF TABLE
LS BUFFER TABLE
1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*
17*18*19*20*21*22*23*24*25*26*27*28*29*30*31*32*
33*34*35*36*37*38*39*40*41*42*43*44*45*46*47*48*
49*50*51*52*53*54*55*56*57*58*59*60*61*62*63*64*
65*66*67*68*69*70*71*72*73*74*75*76*77*78*79*80*
81*82*83*84*85*86*87*88*89*90*91*92*93*94*95*96*
97*98*99*100*101*102*103*104*105*106*107*108*109*110*111*112*
113*114*115*116*117*118*119*120*121*122*123*124*125*126*127*128*
END OF TABLE
STATE   SIGNAL  TIME
  1     0       0
  1     257     0
```

Figure 4. BDACS on-line printout of the method file.

METHOD FILE

1	258	0
1	259	0
1	265	0
1	266	0
1	267	0
1	297	1000
1	298	2000
1	299	3000
1	300	4000
1	301	5000
1	281	5000
1	302	6000
1	303	7000
1	304	8000
1	305	9000

END

Figure 4. BDACS on-line printout of the method file (Cont'd).

HSB ASSIGNMENT & DATA

BIT#	MNEMONIC NAME
1	HSB BIT #1
2	HSB BIT #2
3	HSB BIT #3
4	HSB BIT #4
5	HSB BIT #5
6	HSB BIT #6
7	HSB BIT #7
8	HSB BIT #8
9	HSB BIT #9
10	HSB BIT #10
11	HSB BIT #11
12	HSB BIT #12
13	HSB BIT #13
14	HSB BIT #14
15	HSB BIT #15
16	HSB BIT #16

Figure 5. BDACS on-line printout of the HSB assignment list and data (partial).

HSB ASSIGNMENT & DATA

TIME(NS)	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
5500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
11000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
16500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
22000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
27500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
33000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
38500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
44000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
49500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
55000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
60500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
66000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
71500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
77000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
82500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
88000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
93500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
99000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
104500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
110000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
115500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
121000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
126500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
132000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
137500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
143000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
148500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
154000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
159500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
165000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
170500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
176000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
181500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
187000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
192500	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
198000	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1

Figure 5. BDACS on-line printout of the HSB assignment list and data (partial (Cont'd).

REDUCED MUX & LSB DATA

TIME(US)	STATE	SIGNAL	MNEMONIC
0	1	29M	MUX CHANNEL #29
0	1	1L	LSB CHANNEL #1
100	1	46M	MUX CHANNEL #46
100	1	2L	LSB CHANNEL #2
200	1	56M	MUX CHANNEL #56
200	1	61M	MUX CHANNEL #61
200	1	62M	MUX CHANNEL #62
200	1	3L	LSB CHANNEL #3
300	1	79M	MUX CHANNEL #79
300	1	4L	LSB CHANNEL #4
400	1	93M	MUX CHANNEL #93
400	1	95M	MUX CHANNEL #95
400	1	5L	LSB CHANNEL #5
500	1	110M	MUX CHANNEL #110
500	1	111M	MUX CHANNEL #111
500	1	6L	LSB CHANNEL #6
600	1	125M	MUX CHANNEL #125
600	1	126M	MUX CHANNEL #126
600	1	127M	MUX CHANNEL #127
600	1	7L	LSB CHANNEL #7
700	1	8L	LSB CHANNEL #8
800	1	9L	LSB CHANNEL #9
900	1	10L	LSB CHANNEL #10
1000	1	11L	LSB CHANNEL #11
1100	1	12L	LSB CHANNEL #12
1200	1	13L	LSB CHANNEL #13
1300	1	14L	LSB CHANNEL #14
1400	1	15L	LSB CHANNEL #15
1500	1	16L	LSB CHANNEL #16
1600	1	17L	LSB CHANNEL #17
1700	1	18L	LSB CHANNEL #18
1800	1	19L	LSB CHANNEL #19

Figure 6. BDACS on-line printout of MUX and LSB data lines (partial).

REDUCED MUX & LSB DATA

1900	1	20L	LSB CHANNEL #20
2000	1	21L	LSB CHANNEL #21
2100	1	22L	LSB CHANNEL #22
2200	1	23L	LSB CHANNEL #23
2300	1	24L	LSB CHANNEL #24
2400	1	25L	LSB CHANNEL #25
2500	1	26L	LSB CHANNEL #26
2600	1	27L	LSB CHANNEL #27
2700	1	28L	LSB CHANNEL #28
2800	1	29L	LSB CHANNEL #29
2900	1	30L	LSB CHANNEL #30
3000	1	31L	LSB CHANNEL #31
3100	1	32L	LSB CHANNEL #32
3200	1	33L	LSB CHANNEL #33
3300	1	34L	LSB CHANNEL #34
3400	1	35L	LSB CHANNEL #35
3500	1	36L	LSB CHANNEL #36
3600	1	37L	LSB CHANNEL #37
3700	1	38L	LSB CHANNEL #38

Figure 6. BDACS on-line printout of MUX and LSB data lines (partial) (Cont'd).

11.2 PHAS5.SR Subroutine Code

Many of the subroutines used in PHAS5.SR, especially those associated with file I/O which use RDOS system calls, are identical to those already described in sections 7 and 9 and will not be repeated here. Other subroutines unique to PHAS5.SR are described below.

(a) GTDAT retrieves the reduced data entries for the MUX or LSB data previously stored during PHAS4.SR operations. Initially, the subroutine reads an entire block of 256 data entries into a buffer area pointed to by BUFFR. As succeeding subroutine calls are made, the current entry pointer is updated until all entries in the buffer have been scanned. The buffer is then refilled with new entries and the scanning is continued. Upon exit from the subroutine accumulator AC2 contains the pointer to the current entry being processed.

(b) OPEN retrieves an assignment list entry from from one of the assignment files ASSIGNA.DA, ASSIGNB.DA, or ASSIGNC.DA. The first parameter for the subroutine call contains the point number of the

signal whose assignment mnemonic text is required. The second parameter contains the pointer to the buffer containing part of the associated assignment file. In particular, files ASSIGNA.DA, ASSIGNB.DA, and ASSIGNC.DA are read into buffer areas pointed to by BUFRA, BUFRB, and BUFRC, respectively. These buffers are 256 words long and, thus, contain 32 assignment mnemonic text strings, each being 8 words (16 bytes) long. If the required assignment text string is not among the current set of 32 strings presently in the buffer, the subroutine performs a disk-block read operation using system call .RDB to retrieve the required text string from the appropriate assignment file. A normal exit is made from the subroutine with accumulator AC2 containing a byte pointer to the required text string within the assignment file buffer. An abnormal exit is made to the error routine for all system errors.

(c) PRFLE prints the contents of an ASCII data file on the on-line printout device. The one parameter required for the subroutine is the name pointer to the ASCII data file to be printed. The data file is opened on RDOS channel No. 4; the file is read line by line, and each line is printed using subroutine PRLNE (described next); channel No. 4 is then closed. No exceptional condition is explicitly contained in this subroutine.

(d) PRLNE is the basic code for controlling and directing the on-line printout for BDACS. Two parameters are required for this subroutine. The first parameter contains a byte pointer to the message to be printed. The second parameter contains (in absolute magnitude) the number of lines that the message will require for printing. If the second parameter is a negative number, a top-of-form operation is performed before the printing. A maximum of 30 lines per page may be printed before an automatic top-of-form operation occurs. The subroutine maintains a current line count (LNCNT) and page count (PGCNT), and, when a top-of-form operation is performed, a new page number is printed. The BDACS operator may suppress printing by setting the sense switches on the CPU panel to all "1's." This suppression will continue until either the sense switches are reset (i.e., one or more to "0") or until a programmed top-of-form operation is initiated. The BDACS operator must then reapply the sense switches if suppression of the current phase of the on-line printout is to continue.

(e) CHKPM determines if the on-line printout mask portion of MUX or LSB mask tables contains on-line printout requests. Two parameters are necessary for this subroutine. The first parameter contains the points to the mask table, either MSKTB or LSBTB for the MUX or LSB, respectively. The second parameter contains the offset count for the different parts of the mask table (see sect. 6.2(a) and (b)). Upon entry to the subroutine, accumulator AC0 contains the on-line printout request flag, initially set to zero. If any on-line printout mask word in the mask table is nonzero (indicating that at least one signal requires printing), the accumulator AC0 is incremented.

(f) MVBYT transfers a text string from a source region to a destination region. The one parameter required for this subroutine contains the maximum number of bytes to be transferred. Upon entry to the subroutine, accumulators AC0 and AC2 contain the byte pointers to the destination text string and the source text string, respectively. Transfer of bytes continuses until either a null byte is detected in the source string or else the maximum number of bytes, as specified by the input parameter, has been transferred. Upon exit from the subroutine, accumulator AC2 contains the byte pointer to the byte following the end of the destination text string.

(g) FMSPN creates an ASCII decimal integer text string corresponding to the single-precision unsigned binary integer contained in accumulator AC0. Upon entry to the subroutine, accumulator AC2 contains the byte pointer to the start of the text area where the string is created. Leading zeros in the generated text string are suppressed to blanks. Upon exit from the subroutine, accumulator AC2 contains the byte pointer to the byte following the generated text string.

(h) FMDPN creates an ASCII decimal integer text string corresponding to the double-precision unsigned binary integer contained in accumulators AC0 and AC1. The operation is similar to that in FMSPN.

(i) MTAIN performs a magnetic-tape free format "read" operation from the magnetic-tape file opened on channel No. 3. The single parameter required by the subroutine contains an address to branch to when an "end-of-file" is detected. The data are transferred to the buffer pointed to by BUFFR in blocks of 768 words each. Except for "end-of-file," all system errors cause a branch to the system error routine.

(j) TMPIN retrieves a block of data from the disk file opened on channel No. 3. The single parameter required by this subroutine contains an address to branch to when an "end-of-file" is detected. The data are transferred to the buffers pointed to by BUFFR in blocks of 768 words each. Except for the "end-of-file," all system errors cause a branch to the system error routine.

11.3 PHAS5.SR Name/Text Strings and Buffers

The PHAS5.SR name and text strings are constructed in a manner similar to that described in section 7.3. Particular strings, unique to PHAS5.SR, are those used for "headers" and standard line formats for the on-line printout. The buffer areas used by PHAS5.SR are pointed to by the following parameters.

(a) UTBPT defines a region of 68 words used for utility byte transfer operations.

(b) BUFRA is divided into two parts. The first part contains a 256-word data storage area. The second part is a three-word buffer "header," which has a negative offset from BUFRA. The first word of the "header" (at offset -3) contains the starting signal point number for the HSB. The second word of the "header" (at offset -2) contains the RDOS channel number (in the RH part) and the I/O block count (in the LH part) associated with the HSB assignment file (ASSIGNA.DA). The third word of the "header" (at offset -1) initially contains 177777. During PHAS5.SR operation, this word will contain the block number of the current ASSIGNA.DA block contained in the data part of the buffer. This current block number is updated under control of subroutine OPEN (see sect. 11.2(b)).

(c) BUFRB is similar to BUFRA, except that the present buffer is associated with the MUX assignment file (ASSIGNB.DA).

(d) BUFRC is similar to BUFRA, except that the present buffer is associated with the LSB assignment file (ASSIGNC.DA).

(e) BUFFR contains a data area sufficient to hold 256 LSB or MUX reduced data entries (presently 768 words). Entry blocks of this size are read into the buffer either from magnetic-tape file MT0:6 (if the "M" option is in effect) or from disk file TEMPC.TM, under control of subroutines MTAIN or TMPIN, respectively (see sect. 11.2(i) and (j)).

11.4 PHAS5.SR Error Routine

The error routine for PHAS5.SR operates identically to that for PHAS1.SR (see sect. 7.4). At present, four program errors are monitored:

(a) ERR1--"ASSIGN FILE READ ERROR OCCURRED"--a system error occurred during the "read" operation in subroutine OPEN (see section 11.2(b)).

(b) ERR2--"ASSIGN FILE SEQUENCE ERROR OCCURRED"--the signal point number for a requested assignment file text string was less than the starting-signal number specified in the "header" of the associated assignment file buffer (see sect. 11.3(b)).

(c) ERR3--"LINE COUNT ERROR"--the updated line count in subroutine PRLNE is negative.

(d) ERR4--"MUX OR LSB POINT # OUT OF BOUNDS"--the signal point number contained in a MUX or an LSB reduced data entry is less than the starting-signal point number.

12. OVERLAY MODULE ERMSG.SR

This overlay module contains the part of the control program that performs an orderly closing of BDACS functions and returns control to the RDOS-CLI. The overlay module can be entered from the root binary segment in one of two modes. The "normal" mode is entered when all BDACS control tasks have been successfully performed and a normal exit has been made from the preceding overlay modules. Under these conditions, the error code word (ERCOD, see sect. 6.1(d)) contains zero as a flag to immediately branch to the address NORML in ERMSG.SR. Closeout is then achieved by performing the following tasks.

(a) Reset all RDOS channels using the system call .RESET.

(b) Release the magnetic-tape drive using the system call .RLSE. Ignore any system errors, since the magnetic tape may not have been initialized.

(c) Return to the CLI by using the system call .RTN.

The "exceptional" mode of entering ERMSG.SR occurs when an error has been detected during the operation of an earlier overlay module. The error code word (ERCOD) contains the error code as described in section 7.4.

Two types of errors are recognized: program and system. For a system error, ERCOD contains 177777 as a flag, and the recovery parameter (RECOV, see sect. 6.1(b)) contains the system-generated error code. Closeout during a system error condition is achieved in a manner similar to that described in (a), (b), and (c) above for "normal" closeout. However, instead of an exit to the CLI by the normal return (i.e., .RTN), for the present situation the system error code from RECOV is retrieved and stored in accumulator AC2, and return is made to the CLI using the system call .ERTN.

When a program error occurs, ERCOD contains the program error code as described in section 7.4. For this case the following tasks are performed within ERMSG.SR.

(a) Decompose the program error code into the module number and error number; determine the error message byte pointer associated with the error number and the recoverable error bit.

(b) Type the module message to the output console:

"ERROR IN PHASE n"

where n is replaced by the overlay module number in which the error occurred.

(c) Type the error message describing the associated error number. These messages are discussed in the preceding sections describing the different overlay modules.

(d) Determine if a recoverable error occurred by checking the recoverable error bit processed in (a) above. If the error is recoverable, return to the root-binary segment. Otherwise, close out the BDACS control program in the "normal" mode as described above.

The tables that are used in ERMSG.SR are described below.

(a) TBLPT contains a list of pointers to the beginning of the message pointer tables. One such message-pointer table is associated with each of the preceding overlay modules (i.e., PHAS1, PHAS2, PHAS3, PHAS4, PHAS5).

(b) PHAS1 contains a list of error-message pointers for overlay module PHAS1.SR. A one-to-one correspondence exists between these error-message pointers and the error table entries described in section 7.4.

(c) PHAS2, PHAS3, PHAS4, and PHAS5 are similar to PHAS1 but are associated with overlay modules PHAS2.SR, PHAS3.SR, PHAS4.SR, and PHAS5.SR, respectively. Note: If bit 1B0 is set to "1" for any message pointer, the associated error is considered "recoverable" and is processed as described in (d) above.

The error messages are RDOS ASCII text strings with mode 1 packing. The assembly language listing for ERMSG.SR is shown in appendix A, section A-7.

13. CONTROL PROGRAM SYSTEM PARAMETERS--BDACS.SR

This file contains those BDACS control parameters which may be modified or extended as the BDACS hardware system is changed or augmented to meet future requirements. As mentioned in section 2, BDACS.SR should be assembled along with the appropriate root binary segment or overlay module files to reflect the present value of these system parameters. A brief description of these parameters along with their present value (in octal) is given below, while an assembly language listing for BDACS.SR is given in appendix A, section A-8.

(a) NOP=000401 provides a mnemonic "no operation" instruction (i.e., JMP .+1). This parameter is not expected to change if the system is augmented.

(b) CLK=054 contains the RTC device interrupt code. Note: this clock interrupt code is *not* the standard RDOS code.

(c) MUX=032 contains the multiplexer controller device interrupt code which is hardwire strapped at the controller.

(d) MUXOB=000300 contains the output multiplexer panel buffer address. This address is hardwire strapped within the multiplexer controller.

(e) IMCLK=000004 contains the interrupt mask bit for the RTC.

(f) IMMUX=001000 contains the interrupt mask bit for the multiplexer controller.

(g) IMDKP=000400 contains the interrupt mask bit for the moving-head disk controller.

(h) BDACA=200 contains the number of signal points located on the input MUX panels (128 decimal).

(i) BDACB=100 contains the number of signal points located on the output MUX panel (64 decimal).

(j) BDACD=144 contains the starting cylinder (track) number for the BDACS raw data file BDACS.DA (see sect. 3.6).

(k) BDACE=144 contains the number (100 decimal) of complete cylinders used for file BDACS.DA. Thus, the size of this file is 2400 disk blocks. This size should be reflected in the UFT entry of the RDOS system directory SYS.DR and in the RDOS map directory MAP.DR. If a full initialization of the BDACS disk cartridge is required, the BDACS operator must insure that this file information is properly supplied (e.g., by using the RDOS disk editor DSKED.SV).

(l) BDACF=1 contains the minimum or starting input MUX panel signal number.

(m) BDACG=200 contains the maxium or last input MUX panel signal number (128 decimal).

(n) BDACH=401 contains the minimum or starting output MUX panel signal number (257 decimal).

(o) BDACI=476 contains the maximum or last output MUX panel signal number (318 decimal). Note: this number does not include the two special-purpose control points No. 319 and 320 used to control HSB data-retrieval operations (see sect. 9).

(p) BDACJ=200 contains the maximum number of entries in the output control list (128 decimal).

(q) BDACK=03 contains the number of words per entry in the output control list.

(r) BDACL=03 contains the number of words per entry in the MUX and LSB reduced data file entry.

(s) BDACM=6000 contains the number of words per data block used during data acquisition and reduction (3072 decimal).

(t) BDACN=144 contains the maximum number of sequence errors allowed (100 decimal).

(u) BDACO=4000 contains the number of words in the HSB unit (2048 decimal).

(v) BDACP=200 contains the number of signal points located on the LSB unit (128 decimal).

(w) BDACQ=1 contains the minimum or starting signal number for the LSB.

(x) BDACR=200 contains the maximum or last signal number for the LSB (128 decimal).

(y) BDACS=1 contains the minimum or starting bit number for the HSB unit.

(z) BDACT=20 contains the maximum or last bit number for the HSB unit (16 decimal). Note: the number of bits per word in the HSB cannot be changed from the present value of 20_8 (16 decimal) without major changes in the overlay module responsible for HSB data retrieval.

(aa) BDACU=60 contains the minimum MUX sample rate (in μs, 50 decimal).

(bb) BDACV=7777 contains the maximum MUX sample rate (in μs, 4095 decimal).

(cc) BDACW=764 contains the minimum HSB sample rate (in ns, 500 decimal).

(dd) BDACX=23420 contains the maximum HSB sample rate (in ns, 10,000 decimal).

(ee) BDAC1=10 contains the number of 16-bit input MUX words (8 decimal).

(ff) BDAC2=4 contains the number of 16-bit output MUX words (4 decimal).

(gg) BDAC3=10 contains the number of 16-bit LSB words (8 decimal).

(hh) BDAC5=600 contains the number of words required for the MUX output control list buffer (384 decimal).

(ii) BDAC6=20 contains the number of words offset between the start of the MUX or LSB mask table and the on-line printout part of the table (16 decimal).

(jj) BDAC7=7 contains the number of input MUX low-speed words (7 decimal).

(kk) BDAC8=3000 contains the number of high-speed/low-speed word pairs per raw data block (1536 decimal).

(ll) BDAC9=1400 contains the number of words in the MUX and LSB reduced data block (768 decimal).

14. RECOMMENDATIONS FOR FUTURE SOFTWARE AUGMENTATION

A major shortcoming of the present BDACS control program is the inability of the program to provide a real-time analysis of and response to the incoming data during data acquisition. Presently, the control functions to be provided by the program are prearranged and occur independently of the data being monitored.

It is possible to provide real-time data analysis/response for BDACS. However, several basic adjustments and changes to the BDACS software package would be required. These changes are outlined in the following paragraghs.

(a) Presently, during PHAS2.SR data acquisition and control, program control switches between the interrupt service routine (when a device interrupt occurs) and the control loop which continually monitors and updates the control output list for the next control point change (see sect. 8.2 and 8.3). If the real-time analysis/response option is to be implemented, then an additional branch must be provided to the program code controlling this analysis/response action.

(b) A specific example and possible implementation:

EXAMPLE

● Allow a single low-speed word to be available for analysis. Thus, the individual bits or a pattern of several bits in combination within this word will be recognized as a request for specific control or response actions.

● Let the 1B0 bit for the word be strapped as a "1." Thus, when the word is monitored and stored in the memory buffer, bit 1B0 will be set.

● Provide a circular buffer pointer which will point to the current word to be analyzed and, upon updating, will point to the succeeding word.

● When a current word has been retrieved for analysis, the storage area in memory will be reset.

● With the current word as data, a branch is made to the analysis/response control subroutine for appropriate action. Upon return from this subroutine, a control loop is entered and scanning is continued until the word is again monitored and becomes "set."

● The interrupt service routine would be essentially unchanged, except, upon exit from the RTC service, a special branch to the predefined control operations control loop would be made. The exit would then be from this loop to the control loop specified above.

(c) In the example, timing considerations must be made to insure proper operation. In particular, the analysis/response subroutine must be efficiently constructed so that the words to be analyzed do not "stack up." This condition may also require the minimum allowable sample rate to be longer than the present minimum (50 μs).

(d) The particulars of the analysis/response subroutine depend in detail upon the specific real-time control tasks to be performed. These tasks may possibly change from test to test and, if this occurs, either several control programs must be individually assembled and linked or several separate PHAS2.SR types of overlay modules must be assembled and linked in a universal control program. If the latter choice is made, additional provisions must be made in the root-binary segment of the universal control program to choose one of the several PHAS2.SR overlay modules to load in at runtime.

(e) One final modification to the present control program may be required. Presently, the BDACS control program resides in memory with the RDOS executive program. The controlling factor in the expansion of the BDACS control program is primarily the size of the PHAS2.SR overlay module, since this module contains the twin data blocks--each 6000_8 words in length. If the PHAS2.SR module must be greatly expanded to

include the analysis/response subroutine code, an incompatibility may arise in which the core storage is insufficient to contain both the BDACS control program and RDOS executive program. Since PHAS2.SR does not require any RDOS system calls, if this incompatibility arises, the executive program can be temporarily transferred to disk, and the core storage used by the executive may be made available to the control program. Before an exit from PHAS2.SR, the executive program would, of course, be returned to core storage. If this action is necessary, it is recommended that tracks No. 201, 202, and 203 (decimal) on the disk be reserved in the RDOS MAP.DR for the executive program storage.

APPENDIX A.--ASSEMBLY-LANGUAGE LISTING FOR BDACS CONTROL PROGRAM

This appendix contains the assembly-language listing for the BDACS control program. The control program is composed of a root-binary section, six overlay modules, and a table of system parameters. Each of these parts is presented in the following sections of the appendix.

APPENDIX A

A-1. Root Binary Section--MONTR.SR

```
NAME BLOCK   NAME=  MONTR.SR

TIME BLOCK

        .TITL   MONTR   ;JCI    6 FEB 76

        .TXTM   1

        .ENT    OVRTN MAGFG SMPRT BLKCT HLDCT
        .ENT    MSKTB CTLTB ERRTN ERCOD HSMON
        .ENT    LSMON LSBTB PRINT RECOV
        .ENT    MEWPT MEBPT

        .EXTN   OVST1 OVST2 OVST3 OVST4 OVST5
        .EXTN   OVST6

        .ZREL

OVRTN:  0               ;OVERLAY RETURN ADDRS
RECOV:  0               ;RECOVERABLE ERROR RETURN
ERRTN:  ERROR           ;ERROR RETURN
ERCOD:  0               ;ERROR CODE
MAGFG:  0               ;MAG TAPE FLAG
PRINT:  0               ;PRINT FLAG
SMPRT:  0               ;SAMPLE RATE
HSMON:  0               ;HS BUFFER FLAG
LSMON:  0               ;LS BUFFER MONITOR
BLKCT:  BDACE*2         ;DATA BLOCK COUNT
HLDCT:  0               ;HOLDOFF COUNT

        .REL

MSKTB:  .BLK    BDAC1*3 ;DATA INPUT MASK BITS
                0
LSBTB:  .BLK    BDAC3*3 ;LS BUFFER TABLE
                0
CTLTB:  .BLK    BDAC5+3 ;CONTROL LIST TABLE
                0

OVJMP:  0               ;OVERLAY JUMP ADDRESS
OVLOD:  LDA     0,0,3   ;NODE/REGION
        INC     3,3
        LDA     2,0,3   ;START ADDRS
        INC     3,3
        STA     2,OVJMP ;JUMP ADDRESS
        STA     3,OVRTN
        ADC     1,1     ;UNCONDITIONAL LOAD
        .SYSTM
        .OVLOD  7       ;LOAD IN OVERLAY
        JMP @   ERRTN   ;ERROR
        JMP @   OVJMP   ;ENTER OVERLAY
```

```
START:  LDA     0,OLNAM ;'MONTR.OL'
        .SYSTM
        .OVOPN  7       ;OPEN OVERLAY FILE
        JMP @   ERRTN   ;ERROR
        JSR     OVLOD   ;LOAD OL#1
                0
                OVST1   ;START ADDRS
        JSR     OVLOD   ;LOAD OL#2
                1
                OVST2   ;START ADDRS
        JSR     OVLOD   ;LOAD OL#3
                2
                OVST3   ;START ADDRS
        JSR     OVLOD   ;LOAD OL#4
                3
                OVST4   ;START ADDRS
        JSR     OVLOD   ;LOAD OL#5
                4
                OVST5   ;START ADDRS
ERROR:  LDA     0,ERCOD ;ERROR CODE
        COM #   0,0,SZR ;SKIP IF SYSTEM ERROR
        LDA     2,OVRTN ;SET UP RECOVERABLE RETURN
        STA     2,RECOV ;HOLD RECOVERY ADDRS OR SYSTEM ERROR
        JSR     OVLOD   ;LOAD OL#6
                5
                OVST6   ;START ADDRS
        HALT

OLNAM:  .+1*2
        .TXT    /MONTR.OL/

MEWPT=  .               ;METHOD FILE WORD POINTER
MEBPT=  .*2             ;METHOD FILE NAME POINTER
        .BLK    7

        .END    START
```

APPENDIX A

A-2. Overlay Module No. 1--PHAS1.SR

```
NAME BLOCK   NAME=  PHAS1.SR

TIME BLOCK

        .TITL   PHAS1   ;JCI     19 FEB 76

        .TXTM   1

        .ENT    OVST1

        .EXTD   MAGFG SMPRT OVRTN ERRTN ERCOD
        .EXTD   HSMON LSMON PRINT

        .EXTN   MSKTB LSBTB CTLTB MEWPT MEBPT

        .NREL

PNTRI:  MSKTB           ;REPORTED POINTS TABLE POINTER
CNTRI:  BDAC1*3         ;TABLE SIZE
PNTRL:  LSBTB           ;LS BUFFER TABLE POINTER
CNTRL:  BDAC3*3         ;TABLE SIZE
DMADB:  MUXOB           ;MUX OUTPUT BUFFER POINTER
CNTRB:  BDAC2           ;OUTPUT BUFFER LENGTH

SWCHD:  000030          ;DEFAULT SWITCHES L&M
SWCHL:  000020          ;SWITCH 'L'
SWCHM:  000010          ;SWITCH 'M'
SWCHT:  010000          ;SWITCH 'T'

SPLDO:  SPLDS           ;DISABLE SPOOLING
OPFLO:  OPFLE           ;OPEN A CHANNEL
CLFLO:  CLFLE           ;CLOSE A CHANNEL
INMTO:  INMTA           ;INITIALIZE MTO
RDLUO:  RDLUT           ;READ INTI UTILITY
RDSUO:  RDSUT           ;READ SEQUENTIALLY
MVWUO:  MVWUT           ;MOVE WORDS FROM UTILITY
CKNBO:  CKNBR           ;CHECK A #

OVST1:  ADC     0,0     ;FORCE -1
        LDA     2,DMAOB ;OUTPUT BUFFER POINTER
        STA     0,0,2   ;RESET OUTPUT BUFFER
        INC     2,2
        DSZ     CNTRB
        JMP     .-3
        SUB     0,0     ;CLEAR AC0
        LDA     2,PNTRI ;INPUT DATA MASK TABLE
        STA     0,0,2   ;CLEAR THE TABLE
        INC     2,2
        DSZ     CNTRI
        JMP     .-3
        LDA     2,PNTRL ;LS BUFFER MASK TABLE
        STA     0,0,2   ;CLEAR THE TABLE
        INC     2,2
```

```
        DSZ     CNTRL
        JMP     .-3

        JSR @   SPLDO   ;DISABLE SPOOLING
                NAMEF   ;'$TTO'
        JSR @   SPLDO   ;DISABLE SPOOLING
                NAMEG   ;'$LPT'
        JSR @   OPFLO
                NAMEF   ;OPEN $TTO
                01      ;ON CH#1
        JSR @   OPFLO
                NAMEA   ;OPEN COM.CM
                04      ;ON CH#4
        JSR @   RDLUO   ;READ 'MONITOR'
        JSR @   RDSUO   ;READ GLOBAL SWITCHES
                04      ;4 BYTES
        JSR @   MVWUO   ;MOVE THE SWITCHES
                MAGFG   ;TO ROOT BINARY
                02      ;2 WORDS
        JSR @   RDLUO   ;READ METHOD FILENAME
        JSR @   MVWUO   ;MOVE THE FILENAME
                MEWPT   ;TO NAME AREA IN ROOT BINARY
                07      ;7 WORDS (MAX)
        JSR @   CLFLO
                04      ;CLOSE CH#4
        LDA     0,MAGFG ;GLOBAL SWITCH #1
        LDA     1,PRINT ;GLOBAL SWITCH #2
        ADD #   1,0,SNR ;SKIP IF SWITCHES PRESENT
        LDA     0,SWCHD ;USE DEFAULT SWITCHES
        LDA     2,SWCHM ;SWITCH MASK 'M'
        AND     0,2     ;MASK FOR MAG TAPE
        STA     2,MAGFG ;SET MAG TAPE FLAG
        LDA     2,SWCHL ;SWITCH MASK 'L'
        AND #   2,0,SNR ;SKIP IF $LPT REQUIRED
        JMP     .+3
        SUBZL   0,0     ;FORCE 1B15
        JMP     .+4
        LDA     0,SWCHT ;SWITCH MASK 'T'
        AND     1,0,SZR ;SKIP IF NO $TTO REQUIRED
        SUBZR   0,0     ;FORCE 1B0
        STA     0,PRINT ;SET PRINT FLAG
        LDA     0,MAGFG ;MAG TAPE FLAG
        MOV #   0,0,SZR ;SKIP IF NO MAG TAPE
        JSR @   INMTO   ;INITIALIZE MTO

        JSR @   OPFLO   ;OPEN
                MEBPT   ; METHOD FILE
                04      ;ON CH#4
        JSR @   RDLUO   ;READ IDENTIFICATION
        JSR @   RDLUO   ;READ SAMPLE RATE
        LDA     2,UTBP1 ;UTILITY BYTE POINTER
        JSR @   GTSP1   ;FORM SP #
        JSR @   CKNBO   ;CKECK SR
ERR10:          BDACU   ;LOWER BOUND
                BDACV   ;UPPER BOUND
        STA     0,SMPRT ;STORE IN ROOT BINARY
        JSR @   RDLUI   ;READ DURATION TIME
```

APPENDIX A

```
         LDA     2,UTBP1 ;UTILITY BYTE POINTER
         JSR @   GTDP1   ;FORM DP #
         STA     0,DURAT ;HOLD THE #
         STA     1,DURAT+1
         JSR @   RDLU1   ;READ HS BUFFER FLAG
         LDA     2,UTBP1 ;UTILITY BYTE POINTER
         JSR @   GTSP1   ;FORM SP #
         MOV #   0,0,SNR ;SKIP IF NOT 0
         JMP     JP9
         JSR @   CKNB0   ;CHECK HSB SR
ERR12:           BDACW   ;LOWER BOUND
                 BDACX   ;UPPER BOUND
         STA     0,HSMON ;STORE IN ROOT BINARY
         JSR @   GTBY1   ;GET NEXT BYTE
         LDA     1,ASCAK ;<*>
         SUB     2,2     ;CLEAR AC2
         SUBZ    0,1,SNR ;SKIP IF NOT <*>
         MOVR    2,2     ;SET 1B0
         LDA     0,HSMON ;HS BUFFER FLAG
         ADD     2,0     ;ADD IN PRINT FLAG
         STA     0,HSMON
JP9:     JSR @   RDLU1   ;READ LS BUFFER FLAG
         LDA     2,UTBP1 ;UTILITY BYTE POINTER
         JSR @   GTSP1   ;FORM SP #
         JSR @   CKNB1   ;CHECK THE  #
ERR11:           BDACF   ;LOWER BOUND
                 BDACF+17;UPPER BOUND
         STA     0,LSMON ;STORE IN ROOT BINARY

         JSR     TABLE   ;CONSTRUCT MONITOR POINT TABLE
                 MSKTB   ;TABLE POINTER
                 BDACA   ;LIST SIZE
                 BDAC1*2 ;OFFSET SIZE
         JSR     TABLE   ;CONSTRUCT LS BUFFER TABLE
                 LSBTB   ;TABLE POINTER
                 BDACP   ;LIST SIZE
                 BDAC3*2 ;OFFSET SIZE
         JMP     JP0     ;CONTINUE

TBRTN:   0               ;RETURN ADDRS
TBPNT:   0               ;TABLE POINTER
TBLSZ:   0               ;TABLE LIST SIZE
TBOFF:   0               ;TABLE OFFSET SIZE
TABLE:   LDA     0,0,3   ;POINTER
         STA     0,TBPNT
         LDA     0,1,3   ;LIST SIZE
         STA     0,TBLSZ
         LDA     0,2,3   ;OFFSET
         STA     0,TBOFF
         STA     3,TBRTN ;RETURN ADDRS
         JSR @   RDLU1   ;READ REPORTED POINTS HEADER
LP1:     JSR @   RDLU1   ;READ A LINE OF REPORTED POINTS
         LDA     2,UTBP1 ;UTILITY BYTE POINTER
         JSR @   GTSP2   ;FORM SP #
```

```
                TBEND   ;NO # FOUND, END OF TABLE
LP2:    NEG     0,0     ;DECREMENT # BY -1
        COM     0,0
        MOVL #  0,0,SZC ;SKIP IF # POSITIVE
        JSR @   ERRO1   ;ERROR, OUT OF BOUNDS
ERR1:   LDA     1,TBLSZ ;# OF INPUT POINTS
        SUBZ #  1,0,SZC ;SKIP IF # <= MAX
        JSR @   ERRO1   ;ERROR OUT OF BOUNDS
ERR2:   JSR @   FMMSI   ;FORM MASK AND DISP.
        LDA     3,TBPNT ;START OF INPUT DATA MASKS
        ADD     0,3     ;ADD IN WORD OFFSET
        STA     1,TEMP1 ;HOLD THE MASK BIT
        STA     3,TEMP2 ;HOLD THE POINTER
        LDA     0,0,3   ;MASK WORD
        AND #   1,0,SNR ;SKIP IF ALREADY PRESENT
        ADD     1,0     ;ADD IN NEW BIT
        STA     0,0,3   ;RESTORE THE WORD
        JSR @   GTBY1   ;GET BYTE FOLLOWING THE #
        LDA     1,ASCCR ;<CR>
        SUB #   0,1,SNR ;SKIP IF NOT EOL
        JMP     LP1
        LDA     1,ASCAK ;<^>
        SUB #   0,1,SZR ;SKIP IF LPT FLAG
        JMP     LP3     ;GET NEXT #
        LDA     1,TEMP1 ;RETRIEVE MASK BIT
        LDA     3,TEMP2 ;RETRIEVE POINTER
        LDA     0,TBOFF ;TABLE OFFSET
        ADD     0,3     ;OFFSET THE POINTER
        LDA     0,0,3   ;MASK WORD
        AND #   1,0,SNR ;SKIP IF ALREADY PRESENT
        ADD     1,0     ;ADD IN NEW BIT
        STA     0,0,3   ;RESTORE THE MASK WORD
LP3:    JSR @   GTSP2   ;FORM SP #
                LP1     ;NO # FOUND, READ NEXT LINE
        JMP     LP2     ;LOOP BACK
TBEND:  LDA     3,TBRTN ;RETURN ADDRS
        JMP     3,3     ;RETURN

HDTIM:  0               ;TIME STORAGE
        0
DURAT:  0               ;DURATION TIME
        0

UTBP1:  UTBPT           ;UTILITY BUFFER POINTER
MVWU1:  MVWUT           ;MOVE WORDS FROM UTILITY
FMMS1:  FMMSK           ;FORM A MASK AND DISP.
RDSU1:  RDSUT           ;READ SEQUENTIALLY INTO UTILITY
RDLU1:  RDLUT           ;READ A LINE INTO UTILITY
RDLU2:  RDLUR           ;READ A LINE, EXIT ON EOF
CKNB1:  CKNBR           ;CKECK A NUMBER
GTSP1:  GTSPN           ;FORM A SP #
GTSP2:  GTSPR           ;FORM A SP #, EXIT IF NONE
GTDP1:  GTDPN           ;FORM A DP #
GTBY1:  GTBYT           ;GET A BYTE
OPFL1:  OPFLE           ;OPEN A CHANNEL
CLFL1:  CLFLE           ;CLOSE A CHANNEL
ERRO1:  ERROR           ;ERROR ROUTINE
```

APPENDIX A

```
TEMP1:  0                   ;TEMPORARY
TEMP2:  0
ASCCR:  015                 ;<CR>
ASCAK:  052                 ;<*>
ASCBK:  040                 ;< >
ASCCN:  136                 ;<¬>

JP2:    ADC     0,0         ;FORCE A -1
        STA     0,TEMP2     ;SET END OF LIST FLAG
        LDA     0,DURAT     ;RETRIEVE DURATION TIME
        LDA     1,DURAT+1
        JMP     JP3         ;STORE DURATION AS FINAL ENTRY
JP0:    JSR @   RDLU1       ;READ CONTROL TABLE HEADER
        JSR @   RDLU2       ;READ CONTROL LINE
                JP2         ;EOF
        LDA     2,UTBP1     ;UTILITY BYTE POINTER
        JSR @   GTSP2       ;FORM SP #
                JP2         ;NO # FOUND
        DSZ     CNTRO       ;SKIP IF OUTPUT TABLE FULL
        JMP     .+2
        JSR @   ERRO1       ;ERROR ROUTINE
ERR3:   STA     0,TEMP1     ;HOLD THE STATE
        JSR @   GTSP1       ;FORM SP #
        STA     0,TEMP2     ;HOLD THE SIGNAL #
        JSR @   GTDP1       ;FORM DP #
JP3:    LDA     2,HDTIM     ;PREVIOUS TIME
        LDA     3,HDTIM+1
        STA     0,HDTIM     ;STORE PRESENT TIME
        STA     1,HDTIM+1
        SUBZ    3,1,SZC     ;DELTA TIME
        SUBZ    2,0,SKP
        ADC     2,0
        MOV #   0,0,SNC     ;SKIP IF OLD<=NEW TIME
        JSR @   ERRO1       ;ERROR ROUTINE
ERR4:   MOV #   0,0,SZR     ;SKIP IF NO OVERFLOW
        JSR @   ERRO1       ;ERROR ROUTINE
ERR5:   STA @   1,PNTRO     ;STORE DELTA TIME
        ISZ     PNTRO       ;INC THE POINTER
        LDA     0,TEMP2     ;RETRIEVE SIGNAL #
        ADC     1,1         ;FORCE MASK TO -1
        MOV #   0,0,SNR     ;SKIP IF NOT ACQUISITION HOLDOFF
        JMP     JP1
        COM #   0,0,SNR     ;SKIP IF NOT END OF LIST
        JMP     JP1
        LDA     1,CTLOF     ;CONTROL # OFFSET (257)
        SUBZ    1,0,SNC     ;REMOVE OFFSET, SKIP IF IN BOUNDS
        JSR @   ERRO1       ;ERROR ROUTINE
ERR6:   LDA     1,CTLMX     ;# OF CONTROL POINTS
        SUBZ #  1,0,SZC     ;SKIP IF IN BOUNDS
        JSR @   ERRO1       ;ERROR ROUTINE
ERR7:   JSR @   FMMS1       ;FORM MASK AND DISP.
        COM     1,1         ;COMPLEMENT THE MASK
        MOV #   0,0,SNR     ;SKIP IF NOT 0 DISP.
        LDA     0,BUFOF     ;ELSE FORCE MAX DISP.
        LDA     2,PNTRB     ;OUTPUT BUFFER POINTER
        ADD     0,2         ;ADD IN DISP.
        LDA     0,TEMP1     ;RETRIEVE STATE
```

```
        MOVZR   0,0       ;SHIFT LS BIT TO CARRY
        MOVL    2,0       ;SET STATE BIT IN POINTER
JP1:    STA @   1,PNTR0   ;STORE MASK
        ISZ     PNTR0     ;INC THE POINTER
        STA @   0,PNTR0   ;STORE STATE BIT AND POINTER
        ISZ     PNTR0     ;INC POINTER
        COM #   0,0,SZR   ;SKIP IF END OF LIST
        JMP     JP0+1     ;LOOP BACK

        JSR @   CLFL1
                04        ;CLOSE CH#4
        JSR @   OPFL1     ;OPEN
                NAMEE     ;$TTI
                02        ;ON CH #2
JP10:   JSR @   CREA1     ;CREATE A FILE
                NAMED     ;'TEMPA.TM'
        JSR @   OPFL1
                NAMEC     ;OPEN PREAM.DA FILE
                04        ;ON CH#4
        JSR @   OPFL1
                NAMED     ;OPEN TEMPA.TM FILE
                03        ;ON CH#3

LP4:    JSR @   RDLU2     ;READ A QUERY LINE
                JP6       ;EOF
        STA     2,TEMP1   ;HOLD THE BYTE POINTER
        LDA     2,UTBP1   ;UTILITY BYTE POINTER
        JSR @   GTBY1     ;GET A BYTE
        LDA     1,ASCAK   ;<*>
        LDA     2,TEMP1   ;RETRIEVE THE POINTER
        SUB     0,1       ;SET FLAG TO 0 IF <*>
        STA     1,TEMP1   ;HOLD THE FLAG
        MOV #   1,1,SNR   ;SKIP IF NOT <*>
        JMP     JP4
        NEG     2,2       ;DECREMENT BY -1
        COM     2,2
        STA     2,TEMP2   ;HOLD THE POINTER
        SUB     0,0       ;CLEAR AC0 TO NUL
        JSR @   STBY1     ;REPLACE <CR> WITH <NUL>
JP4:    JSR @   TYLU1     ;ECHO THE LINE
        LDA     1,TEMP1   ;RETRIEVE THE FLAG
        MOV #   1,1,SNR   ;SKIP IF RESPONSE REQUIRED
        JMP     JP5       ;HOLD THE LINE
        LDA     2,TEMP2   ;RETRIEVE THE POINTER
LP5:    JSR @   RDLU3     ;READ INTO UTILITY
                02        ;FROM CH#2
        ADCZL   0,0       ;FORCE A -2
        ADD     0,2       ;BACK UP
        STA     0,TEMP1   ;STORE -2 AS FLAG
        JSR @   GTBY1     ;GET A BYTE
        LDA     1,ASCCN   ;<¬>
        SUB #   0,1,SZR   ;SKIP IF <¬>
        JMP     JP5
        LDA     0,ASCCR   ;<CR>
        JSR @   STBY1     ;REPLACE <¬> WITH <CR>
        ISZ     TEMP1     ;INC THE FLAG
```

APPENDIX A

```
JP5:    JSR @   WTLU1   ;WRITE THE LINE
        LDA     2,UTBP1 ;UTILITY BYTE POINTER
        ISZ     TEMP1   ;INC THE FLAG, SKIP IF <¬>
        JMP     LP4     ;READ NEXT QUERY
        JMP     LP5     ;CONTINUE QUERY RESPONSE

PNTRB:  MUXOB-1         ;POINTER TO DMA OUTPUT BUFFER
PNTRO:  CTLTB           ;POINTER TO CONTROL LIST
CNTRO:  BDAC5           ;MAX SIZE OF LIST
CTLOF:  BDACH           ;OFFSET TO RUNNING CONTROL POINT #
CTLMX:  BDAC?           ;# OF CONTROL POINTS
BUFOF:  BDAC4           ;MUX OUTPUT BUFFER SIZE

CREA1:  CREAT           ;CREATE A FILE
XFER1:  XFERF           ;TRANSFER ASCII FILES
XBLK1:  XBLK            ;TRANSFER FILES BY BLOCKS
WTLU1:  WTLUT           ;WRITE A LINE FROM UTILITY BUFFER
TYLU1:  TYLUT           ;TYPE A LINE FROM  UTILITY BUFFER
STBY1:  STBYT           ;STORE A BYTE
CLFL2:  CLFLE           ;CLOSE A CHANNEL
TYPM1:  TYPMG           ;TYPE A MESSAGE
UTBP2:  UTBPT   ;UTILITY BYTE POINTER
RDLU3:  RDLUA           ;READ A LINE
GTBY2:  GTBYT           ;GET A BYTE

ASCIN:  116             ;<N>
ASCIY:  131             ;<Y>

JP6:    JSR @   CLFL2
                04      ;CLOSE CH#4
        JSR @   CLFL2
                03      ;CLOSE CH#3
JP11:   JSR @   TYPM1   ;TYPE A MESSAGE
                MSG02   ;MESSAGE #2
        LDA     2,UTBP2 ;UTILITY
        JSR @   RDLU3   ;READ A LINE
                02      ;FROM CH #2
        LDA     2,UTBP2 ;UTILITY
        JSR @   GTBY2   ;GET A BYTE
        LDA     1,ASCIN ;<N>
        SUB #   0,1,SNR ;SKIP IF NOT <N>
        JMP     JP10
        LDA     1,ASCIY ;<Y>
        SUB #   0,1,SZR ;SKIP IF <Y>
        JMP     JP11
        JSR @   CLFL2
                02      ;CLOSE CH#2
        LDA     0,MAGFG ;MAG TAPE FLAG
        MOV #   0,0,SNR ;SKIP IF MAG TAPE
        JMP     JP7
        JSR @   XFER1   ;TRANSFER
                NAMED   ;TEMPA.TM TO
                NAMEH   ;MTC:0
        JSR @   XFER1   ;TRANSFER
                METPT   ;METHOD FILE TO
                NAMEI   ;MTC:1
```

```
        JSR @   XBLK1   ;TRANSFER BLOCK
                NAMEM   ;'ASSIGNA.DA'
                NAMEJ   ;'MTO:2'
        JSR @   XBLK1   ;TRANSFER BLOCK
                NAMEN   ;INPUT MUX ASSIGN FILE TO
                NAMEK   ;MTO:3
        JSR @   XBLK1   ;TRANSFER BLOCKS
                NAMEO   ;LOW SPEED BUFFER ASSIGN FILE TO
                NAMEL   ;'MTO:4'
JP7:    READS   0       ;READ SWITCHES
        MOV #   0,0,SNR ;SKIP IF NOT ALL DOWN
        JMP     JP8
        JSR @   TYPMI   ;TYPE A MESSAGE
                MSG01
        SUB     0,0     ;CLEAR AC0
        INC     0,0,SZR ;DELAY
        JMP     .-1
        READS   0       ;READ SWITCHES AGAIN
        MOV #   0,0,SZR ;SKIP WHEN ALL DOWN
        JMP     .-5     ;LOOP BACK
JP8:    JMP @   OVRTN   ;RETURN TO ROOT BINARY


PH1CD:  400             ;PHASE 1 ERROR CODE
ERTBL:  .+1             ;ERROR TABLE POINTER
        ERR1
        ERR2
        ERR3
        ERR4
        ERR5
        ERR6
        ERR7
        ERR8
        ERR9
        ERR10
        ERR11
        ERR12
        -1              ;END OF TABLE
ERROR:  LDA     0,PH1CD ;PHASE 1 ERROR CODE
        LDA @   2,ERTBL ;ERROR TABLE
        ISZ     ERTBL   ;INC TABLE POINTER
        INC     0,0     ;INC COUNT
        COM #   2,2,SNR ;SKIP IF NOT EOT
        JMP     .+3
        SUB #   2,3,SZR ;SKIP IF ERROR FOUND
        JMP     .-6     ;LOOP BACK
        JMP     .+2
SYSER:  ADC     0,0     ;SYSTEM ERROR FLAG
        STA     0,ERCOD ;STORE IN ROOT BINARY
        JMP @   ERRTN   ;ERROR RETURN TO ROOT BINARY

CKNBR:  LDA     1,0,3   ;LOWER BOUND
        SUBZ #  1,0,SNC ;SKIP IF OK
        JMP     ERROR
        LDA     1,1,3   ;UPPER BOUND
        SUBZ #  0,1,SNC ;SKIP IF OK
        JMP     ERROR
        JMP     2,3     ;NORMAL RETURN
```

APPENDIX A

```
C11:    11
C12:    12
CRRTN:  0                       ;RETURN ADDRESS
CREAT:  LDA     0,0,3           ;NAME POINTER
        INC     3,3
        STA     3,CRRTN         ;STORE RETURN
CREAA:  .SYSTM
        .CREA                   ;CREATE SEQ. FILE
        JMP     .+2             ;ERROR
        JMP @   CRRTN           ;RETURN
        LDA     1,C11
        SUB #   1,2,SZR         ;SKIP IF FILE EXISTS
        JMP @   SYSEI           ;ERROR
        JSR     DELEA           ;DELETE THE FILE
        JMP     CREAA           ;TRY AGAIN

DELET:  LDA     0,0,3           ;NAME POINTER
        INC     3,3
DELEA:  STA     3,USP           ;STORE RETURN
        .SYSTM
        .DELET                  ;DELETE THE FILE
        JMP     .+2
        JMP     0,3             ;NORMAL RETURN
        LDA     1,C12
        SUB #   1,2,SZR         ;SKIP IF NO FILE
        JMP @   SYSEI           ;ERROR
        JMP     0,3             ;NORMAL RETURN

XFRTN:  0                       ;RETURN ADDRESS
XFERF:  LDA     0,0,3           ;SOURCE FILE
        LDA     1,1,3           ;DESTINATION FILE
        STA     0,XFSFL
        STA     1,XFDFL
        STA     3,XFRTN         ;STORE RETURN
        JSR     OPFLE           ;OPEN SOURCE FILE
XFSFL:          0               ;POINTER
                04              ;ON CH#4
        JSR     OPFLE           ;OPEN DESTINATION FILE
XFDFL:          0               ;POINTER
                03              ;ON CH#3
        JSR     RDLUR           ;READ A LINE
                .+3             ;RETURN ON EOF
        JSR     WTLUT           ;WRITE A LINE
        JMP     .-3             ;LOOP BACK
        JSR     CLFLE           ;CLOSE
                04              ;CH#4
        JSR     CLFLE           ;CLOSE
                03              ;CH#3
        LDA     3,XFRTN         ;RETURN ADDRESS
        JMP     2,3             ;RETURN

OPFLE:  LDA     0,0,3           ;NAME POINTER
        LDA     2,1,3           ;CH #
        SUB     1,1             ;USE DEFAULT CHARACTERISTICS
        STA     3,USP           ;STORE RETURN
        .SYSTM
        .OPEN   77              ;OPEN THE CHANNEL
```

```
        JMP @   SYSE1   ;ERROR
        JMP     2,3     ;NORMAL RETURN

CLFLE:  LDA     2,0,3   ;CH #
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .CLOS   77      ;CLOSE THE CHANNEL
        JMP @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

MTANM:  .+1*2
        .TXT    /MT0/
INMTA:  LDA     0,MTANM ;NAME POINTER
        SUB     1,1     ;PARTIAL
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .INIT           ;INITIAL DIRECTORY
        JSR @   SYSE1   ;ERROR
        DIA     0,MTA   ;STATUS
        MOVR #  0,0,SNC ;SKIP WHEN READY
        JMP     .-2     ;WAIT FOR ?REWIND
        JMP     0,3     ;NORMAL RETURN

TYPMG:  LDA     0,0,3   ;MSG POINTER
        INC     3,3
        SUBZL   2,2     ;FORCE A +1 FOR CH#
        JMP     .+5
TYLUT:  SUBZL   2,2     ;FORCE A +1 FOR CH#
        JMP     .+2
WTLUT:  LDA     2,C03   ;CH #3, BY DEFAULT
        LDA     0,UTBPO ;UTILITY BYTE POINTER
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .WRL    77      ;WRITE A LINE
        JMP @   SYSE1   ;ERROR
        JMP     0,3     ;NORMAL RETURN

RDSUT:  LDA     1,0,3   ;BYTE COUNT
        LDA     0,UTBPO ;UTILITY BYTE POINTER
        LDA     2,C04   ;CH #4
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .RDS    77      ;READ SEQUENTIAL BYTES
        JSR @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

SYSE1:  SYSER           ;SYSTEM ERROR
UTBPO:  UTBPT           ;UTILITY BYTE POINTER

C03:    03
C04:    04
C06:    06
C177:   177
RDRTN:  0
RDLUA:  ADC     1,1     ;RETURN FLAG -1
        MOV     2,0     ;BYTE POINTER
        LDA     2,0,3   ;CH #
```

APPENDIX A

```
        INC     3,3
        JMP     .+7
RDLUR:  LDA     1,0,3   ;RETURN ADDRESS
        INC     3,3
        JMP     .+2
RDLUT:  ADC     1,1     ;RETURN FLAG -1
        LDA     0,UTBPO ;UTILITY BYTE POINTER
        LDA     2,C04   ;CH #4
        STA     1,RDRTN ;STORE RETURN
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .RDL    77      ;READ A LINE
        JMP     .+4     ;CHECK FOR EOF
        MOV     0,2     ;MOVE THE BYTE POINTER
        ADD     1,2     ;OFFSET THE POINTER
        JMP     0,3     ;NORMAL RETURN
        LDA     0,C06   ;EOF CODE
        SUB #   0,2,SZR ;SKIP IF EOF
        JMP @   SYSE1   ;ERROR
        LDA     0,RDRTN ;RETURN
        COM #   0,0,SZR ;SKIP IF NO EOF RETURN
        JMP @   RDRTN   ;EOF RETURN
        JMP @   SYSE1   ;ERROR

SPLDS:  LDA     0,0,3   ;DEVICE BYTE POINTER
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .SPDA           ;DISABLE SPOOLING
        JSR @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

MVRTN:  0
MVWUT:  LDA     2,0,3   ;DESTINATION POINTER
        LDA     1,1,3   ;COUNT
        STA     3,MVRTN ;STORE RETURN
        LDA     3,UTBPO ;UTILITY BYTE POINTER
        MOVZR   3,3     ;FORM ADDRESS
        NEG     1,1     ;NEG THE COUNT
        LDA     0,0,3   ;GET WORD
        STA     0,0,2   ;STORE WORD
        INC     3,3
        INC     2,2
        INC     1,1,SZR ;SKIP WHEN DONE
        JMP     .-5
        LDA     3,MVRTN ;RETURN ADDRESS
        JMP     2,3     ;RETURN

OPFL3:  OPFLE           ;OPEN A FILE
CLFL3:  CLFLE           ;CLOSE A CHANNEL
OPMT3:  OPMTA           ;OPEN MAG TAPE FOR FF
RDBL3:  RDBLK           ;READ A DISK BLOCK
MTAW3:  MTAWT           ;WRITE BLOCK TO MTA
MTAS3:  MTASR           ;SPACE REVERSE MTA
MTAE3:  MTAEF           ;WRITE EOF ON MTA
X8RTN:  0               ;RETURN ADDRS
X3LK:   LDA     0,0,3   ;SOURCE FILE
        LDA     1,1,3   ;DESTINATION FILE
```

```
        STA     0,XBLKS ;SOURCE
        STA     1,XBLKD ;DESTINATION
        STA     3,XBRTN ;RETURN ADDRS
        JSR @   OPFL3   ;OPEN
XBLKS:          0       ;DISK SOURCE
                04      ;ON CH #4
        JSR @   OPMT3   ;OPEN MTA
XBLKD:          0       ;DESTINATION
                03      ;ON CH #3
        SUB     0,0     ;CLEAR AC0
        STA     0,XBLKN ;BLOCK #
        JSR @   RDBL3   ;READ DISK BLOCK
XBLKN:          0       ;BLOCK #
                XBEND   ;EOF RETURN
        JSR @   MTAW3   ;WRITE BLOCK TO MTA
        ISZ     XBLKN   ;INC BLOCK #
        JMP     .-5
XBEND:  JSR @   MTAE3   ;WRITE EOF TO MTA
        JSR @   MTAS3   ;SPACE REVERSE MTA
        JSR @   CLFL3   ;CLOSE
                03      ;CH #3
        JSR @   CLFL3   ;CLOSE
                04      ;CH #4
        LDA     3,XBRTN ;RETURN ADDRS
        JMP     2,3     ;RETURN

OPMTA:  LDA     0,0,3   ;NAME POINTER
        LDA     2,1,3   ;CH #
        SUB     1,1     ;DEFAULT CHARACTERISTICS
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .MTOPD  77      ;OPEN MTA FOR FF
        JSR @   SYSE3   ;SYSTEM ERROR
        JMP     2,3     ;NORMAL RETURN

RDBAK:  0               ;RETURN ADDRS
RDBLK:  LDA     1,0,3   ;BLOCK #
        LDA     0,1,3   ;EOF RETURN ADDRS
        STA     0,RDBAK ;STORE ADDRS
        LDA     0,BLKPT ;BLOCK BUFFER POINTER
        LDA     2,BLKCT ;COUNT OF 1
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .RDB    04      ;READ A BLOCK
        JMP     .+2     ;ERROR
        JMP     2,3     ;NORMAL RETURN
        LDA     1,EOFCD ;EOF CODE
        SUB #   1,2,SZR ;SKIP IF EOF
        JSR @   SYSE3   ;SYSTEM ERROR
        JMP @   RDBAK   ;EOF RETURN

BLKPT:  BUFFR           ;BLOCK BUFFER POINTER
BLKCT:  000400          ;BLOCK COUNT
EOFCD:  06              ;EOF ERROR CODE
SYSE3:  SYSER           ;SYSTEM ERROR
SRCMD:  040001          ;SPACE REVERSE COMMAND
EFCMD:  060000          ;EOF COMMAND
WTCMD:  050400          ;WRITE BLOCK COMMAND
EOFMK:  000400          ;EOF MASK
```

APPENDIX A

```
MTASR:  LDA     1,SRCMD ;SPACE REVERSE COMMAND
        JMP     .+2
MTAEF:  LDA     1,EFCMD ;WRITE EOF COMMAND
        JMP     .+3
MTAWT:  LDA     1,WTCMD ;WRITE COMMAND
        LDA     0,BLKPT ;BLOCK BUFFER POINTER
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .MTDIO  03      ;MTA FF
        JMP     .+2     ;ERROR
        JMP     0,3     ;NORMAL RETURN
        LDA     1,EOFMK ;EOF MASK
        AND #   1,2,SNR ;SKIP IF EOF
        JSR @   SYSE3   ;SYSTEM ERROR
        JMP     0,3     ;NORMAL RETURN

FMMSK:  SUBZL   1,1     ;SET BIT TO +1
        MOVZR   0,0,SZC ;SKIP IF NOT 2**0
        MOVZL   1,1     ;SHIFT 1 PLACE LEFT
        MOVZR   0,0,SZC ;SKIP IF NOT 2**1
        ADDZL   1,1     ;SHIFT 2 PLACE LEFT
        MOVZR   0,0,SNC ;SKIP IF     2**2
        JMP     .+3
        ADDZL   1,1     ;SHIFT 4 PLACE LEFT
        ADDZL   1,1
        MOVZR   0,0,SZC ;SKIP IF NOT 2**3
        MOVS    1,1     ;SHIFT 8 PLACE LEFT
        JMP     0,3     ;RETURN

GTDPN:  ADC     0,0     ;DP FLAG
        MOV     0,1     ;ERROR ON NO # FLAG
        JMP     .+6
GTSPR:  LDA     1,0,3   ;EXIT POINTER IF NO #
        INC     3,3
        JMP     .+2
GTSPN:  ADC     1,1     ;ERROR ON NO # FLAG
        SUBZL   0,0     ;SP FLAG
        STA     0,GTFG0 ;SP/DP FLAG
        SUB     0,0     ;CLEAR AC0
        STA     0,GTFG1 ;# FOUND FLAG
        STA     0,GTOVF ;OVERFLOW FLAG
        STA     0,GTSTR ;# STORAGE
        STA     0,GTSTR+1
        STA     1,GTERT ;ERROR EXIT FLAG
        STA     2,GTBPT ;BYTE POINTER
        STA     3,GTRTN ;NORMAL RETURN
        JMP     .+2
GTLP1:  ISZ     GTBPT   ;INC BYTE POINTER
        LDA     2,GTBPT ;GET THE POINTER
        JSR     GTBYT   ;GET THE BYTE
        LDA     1,ASC1C ;<CR>
        SUB     0,1,SNR ;SKIP IF NOT <CR>
        JMP     GTLP2   ;DONE
        LDA     1,ASC19 ;<9>
        ADCZ #  1,0,SZC ;SKIP IF AC1>=AC0
        JMP     GTLP2   ;NOT A DIGIT
        LDA     1,ASC10 ;<0>
```

```
          ADCZ #   0,1,SZC ;SKIP IF AC0>=AC1
          JMP      GTLP2   ;NOT A DIGIT
          ISZ      GTFG1   ;INC # FOUND FLAG
          SUB      1,0     ;FORM BINARY
          STA      0,GTUTL ;HOLD THE #
          LDA      0,GTSTR ;GET PREVIOUS VALUE
          LDA      1,GTSTR+1
          MOVZL    1,1     ;N*2
          MOVL     0,0,SZC ;SKIP IF NO OVERFLOW
          ISZ      GTOVF   ;INC FLAG
          MOVZL    1,3     ;N*4
          MOVL     0,2,SZC ;SKIP IF NO OVERFLOW
          ISZ      GTOVF   ;INC FLAG
          MOVZL    3,3     ;N*8
          MOVL     2,2,SZC ;SKIP IF NO OVERFLOW
          ISZ      GTOVF   ;INC FLAG
          ADDZ     1,3,SZC ;N*10
          INC      0,0
          ADDZ     0,2,SZC ;SKIP IF NO OVERFLOW
          ISZ      GTOVF   ;INC FLAG
          LDA      1,GTUTL ;RETRIEVE THE BINARY
          ADDZ     1,3,SNC ;ADD TO PREVIOUS #
          JMP      .+3
          INCZ     2,2,SZC ;SKIP IF NO OVERFLOW
          ISZ      GTOVF   ;INC FLAG
          STA      2,GTSTR ;HOLD UPDATED #
          STA      3,GTSTR+1
          JMP      GTLP1   ;GET NEXT BYTE
GTLP2:    LDA      3,GTFG1 ;# FOUND FLAG
          MOV #    3,3,SNR ;SKIP IF # FOUND
          JMP      GTLP3
          LDA      0,GTSTR ;RETRIEVE THE #
          LDA      1,GTSTR+1
          LDA      2,GTBPT ;RETRIEVE THE BYTE POINTER
          LDA      3,GTOVF ;OVERFLOW FLAG
          DSZ      GTFGO   ;SKIP IF SP #
          JMP      .+4
          MOV #    0,0,SZR ;SKIP IF NOT SP OVERFLOW
          INC      3,3,SKP ;INC FLAG
          MOV      1,0     ;MOVE SP #
          MOV #    3,3,SZR ;SKIP IF NO OVERFLOW
          JSR @    GTERR   ;ERROR ROUTINE
ERR8:     JMP @    GTRTN   ;NORMAL RETURN
GTLP3:    MOV #    1,1,SZR ;SKIP IF EOL
          JMP      GTLP1   ;GET NEXT BYTE
          LDA      1,GTERT ;ERROR RETURN FLAG
          COM #    1,1,SNR ;SKIP IF ADDRESS SPECIFIED
          JSR @    GTERR   ;ERROR ROUTINE
ERR9:     JMP @    GTERT   ;EOL RETURN, NO # FOUND

GTERR:    ERROR            ;ERROR ROUTINE
GTRTN:    0                ;RETURN ADDRESS
GTERT:    0                ;EOL RETURN FLAG
GTBPT:    0                ;BYPE POINTER STORAGE
GTSTR:    0                ;# STORAGE
          0
GTFGO:    0                ;DP/SP FLAG
```

APPENDIX A

```
GTFG1:  0               ;# FOUND FLAG
GTOVF:  0               ;OVERFLOW FLAG
GTUTL:  0               ;UTILITY
ASCIC:  015             ;<CR>
ASCI9:  071             ;<9>
ASCI0:  060             ;<0>

GTBYT:  LDA     1,BTMSK ;BYTE MASK
        MOVZR   2,2,SNC ;FORM WORD ADDRESS. SKIP IF RHS
        MOVS    1,1     ;SWAP THE MASK
        LDA     0,0,2   ;GET WORD
        AND     1,0,SNC ;MASK THE WORD, SKIP IF RHS
        MOVS    0,0     ;SWAP THE WORD
        MOVL    2,2     ;RESTORE BYTE POINTER
        JMP     0,3     ;RETURN

BTMSK:  377             ;BYTE MASK

STRTN:  0
STBYT:  LDA     1,BTMSK ;BYTE MASK
        AND     1,0     ;MASK THE WORD
        MOVZR   2,2,SNC ;FORM WORD ADDRESS, SKIP IF RHS
        MOVS    0,0,SZC ;SWAP WORD, SKIP IF LHS
        MOVS    1,1     ;SWAP MASK
        STA     3,STRTN ;STORE RETURN
        LDA     3,0,2   ;GET WORD
        AND     1,3     ;MASK THE WORD
        ADD     0,3     ;ADD IN NEW BYTE
        STA     3,0,2   ;RESTORE THE WORD
        MOVL    2,2     ;RESTORE BYTE POINTER
        INC     2,2     ;INC POINTER
        JMP @   STRTN   ;RETURN

        .TXTM   1

NAMEA=  .+2
        .TXT    /COM.CM/

NAMEC=  .+2
        .TXT    /PREAMBLE.DA/

NAMED=  .+2
        .TXT    /TEMPA.TM/

NAMEE=  .+2
        .TXT    /$TTI/

NAMEF=  .+2
        .TXT    /$TTO/

NAMEG=  .+2
        .TXT    /$LPT/

NAMEH=  .+2
        .TXT    /MTO:0/
```

```
NAMEI=  .*2
        .TXT    /MT0:1/

NAMEJ=  .*2
        .TXT    /MT0:2/

NAMEK=  .*2
        .TXT    /MT0:3/

NAMEL=  .*2
        .TXT    /MT0:4/

NAMEM=  .*2
        .TXT    /ASSIGNA.DA/
NAMEN=  .*2
        .TXT    /ASSIGNB.DA/

NAMEO=  .*2
        .TXT    /ASSIGNC.DA/

MSG01=  .*2
        .TXT    /<12>ALL SWITCHES DOWN<12><15>/

MSG02=  .*2
        .TXT    /<12>PREAMBLE FILE OK? (Y,N) /

UTBPT=  .*2
        .BLK    110

BUFFR=  .               ;BLOCK BUFFER
        .BLK    400
        0

        .END
```

APPENDIX A

A-3. Overlay Module No. 2--PHAS2.SR

```
NAME BLOCK   NAME=  PHAS2.SR

TIME BLOCK

        .TITL   PHAS2   ;JCI     23 FEB 76

        .TXTM   0

        .ENT    OVST2

        .EXTD   OVRTN BLKCT SMPRT ERCOD HLDCT CTLTB
        .EXTD   ERRTN

        .NREL

OvST2:  SKPBZ   TTO     ;WAIT FOR TTO IDLE
        JMP     .-1
        NOP
        NOP
        SKPBZ   TTO
        JMP     .-5
        SKPDZ   TTO
        JMP     .-7
        LDA     1,SEEKG ;DISK SEEKING MASK
        DIA     0,DKP   ;DISK STATUS
        AND #   0,1,SZR ;SKIP IF NOT SEEKING
        JMP     .-2
        SKPBZ   DKP     ;WAIT FOR DISK IDLE
        JMP     .-1
        NOP
        NOP
        SKPBZ   DKP
        JMP     .-5
        SKPDZ   DKP
        JMP     .-7
        INTDS           ;DISABLE INTERRUPTS
        LDA     0,1     ;RDOS INT VECTOR
        LDA     1,INTSV ;BDACS INT VECTOR
        STA     0,INTSV ;SAVE RDOS VECTOR
        STA     1,1     ;ENABLE BDACS INT SYSTEM
        LDA     0,INTMK ;BDACS INT MASK
        MSKO    0       ;MASK OUT

        LDA 0   0,SEEK1 ;SEEK COMMAND
        DOAP    0,DKP   ;OUTPUT THE COMMAND
        LDA     1,SEEKD ;SEEK DONE MASK
        DIA     0,DKP   ;GET DKP STATUS
        AND #   0,1,SNR ;SKIP WHEN SEEK FINISHED
        JMP     .-2
        NIOC    DKP     ;CLEAR DISK

        NIOC    MUX     ;RESET THE MUX
        LDA     0,DMUXA ;DMUX I/O START ADDR. AND COUNT
```

```
          DOA     0,MUX
          LDA     0,DMUXB ;DCH INPUT START ADDR.
          DOB     0,MUX
          LDA     0,DMUXC ;HIGH SPEED WORD MASK
          LDA     1,SMPRT ;SAMPLE RATE
          ADD     1,0     ;ADD IN SAMPLE RATE
          DOC     0,MUX
          LDA     0,DMUXD ;DCH INPUT WORD COUNT

          DOCS    0,MUX   ;SET BUSY, RESET DONE

          LDA     2,TTYPT ;MSG POINTER
T1:       LDA     0,0,2   ;MESSAGE WORD
          INCO    2,2     ;INC THE POINTER
T2:       LDA     1,BTMSK ;BYTE MASK
          AND     0,1,SNR ;MASK THE WORD, SKIP IF NUL
          JMP     JUMP1
          DOAS    1,TTO   ;OUTPUT A CHARACTER
          SKPDN   TTO     ;SKIP WHEN DONE
          JMP     .-1
          MOVCS   0,0,SZC ;SWAP THE WORD, SKIP IF VALID
          JMP     T1      ;GET NEXT WORD
          JMP     T2      ;OUTPUT NEXT CHARACTER

BTMSK:    377               ;BYTE MASK
DMUXA:    013003            ;MUX START ADDRESS & COUNT
DMUXB:    BUFS1+1           ;INPUT DMA BUFFER
DMUXC:    140000            ;HIGH SPEED WORD MASK
DMUXD:    100000-BDACM      ;-WORD COUNT, AND 0 IN IBO
DMUXE:    BUFS1             ;INPUT DMA BUFFER (RUNNING)
DMUXF:    100000-BDACM-1    ;-WORD COUNT (RUNNING)
CLKCD:    03                ;IMS RTC CODE
SEEKG:    002000            ;DKP SEEK IN PROGRESS MASK
SEEKD:    040000            ;DKP SEEK DONE MASK
SEEK1:    SEEKC             ;INITIAL SEEK COMMAND
INTMK:    -1-IMCLK-IMMUX-IMDKP    ;BDACS INTERRUPT MASK
INTSV:    INTSR             ;BDACS INTERRUPT SERVICE ROUTINE
TTYPT:    .+1               ;POINTER TO TEXT
          .TXT    /<15><12>READY<12><15><12>/

JUMP1:    NIOC    TTO       ;CLEAR TTO
          LDA     0,CLKCD ;I MS CODE FOR CLOCK
          LDA     1,DMUXE ;DCH INPUT START ADDR. (RUNNING)
          LDA     2,DMUXF ;DCH INPUT WORD COUNT (RUNNING)
          READS   3         ;READ SENSE SWITCHES
          MOVZL # 3,3,SNC ;SKIP WHEN SSO UP
          JMP     .-2
          LDA     3,PNTRO ;OUTPUT TABLE POINTER
          DOAS    0,CLK     ;START THE RTC
          NIOP    MUX       ;START THE MULTIPLEXER
          DOB     1,MUX     ;OUTPUT RUNNING VALUES
          DOC     2,MUX
          INTEN             ;ENABLE INTERRUPTS

L0:       LDA     0,0,3   ;DELTA TIME
          MOV #   0,0,SNR ;SKIP IF NOT 0
          JMP     L1      ;ELSE OUTPUT IMMEDIATELY
```

APPENDIX A

```
        LDA     1,RTCCT ;RTC COUNT
        MOV #   1,1,SNR ;SKIP IF POSITIVE COUNT
        JMP     L0      ;LOOP BACK
        DSZ     RTCCT   ;DEC THE COUNT
        NOP
        DSZ     0,3     ;SKIP IF OUTPUT REQUIRED
        JMP     L0      ;LOOP BACK
L1:     LDA     1,1,3   ;OUTPUT MASK
        LDA     2,2,3   ;STATE AND BUFFER POINTER
        COM #   2,2,SNR ;SKIP IF NOT END
        JMP     L2
        MOV #   2,2,SNR ;SKIP IF NOT END OF HOLDOFF
        JMP     L6
        MOVZR   2,2     ;SET STATE IN CARRY
        LDA     0,0,2   ;GET BUFFER WORD
        AND     1,0     ;MASK THE WORD
        MOV #   0,0,SNC ;SKIP IF +1 STATE
        ADC     1,0     ;RESET THE STATE
        STA     0,0,2   ;RESTORE THE BUFFER WORD
L3:     LDA     0,CTLOF ;CONTROL TABLE OFFSET
        ADD     0,3     ;OFFSET THE POINTER
        JMP     L0      ;LOOP BACK

L6:     ISZ     HLDFG   ;HOLDOFF FLAG
        JMP     L3
L2:     ISZ     ENDFG   ;END OF ACQUISITION/CONTROL FLAG
        JMP     .       ;WAIT FOR LAST DATA BLOCK
        JMP     .-1
L4:     SUB     1,1,SKP ;SET FLAG 0
ERROR:  ADC     1,1,SKP ;SET FLAG -1
        JMP     L5
        LDA     0,PH2CD ;PHASE 2 ERROR CODES
        LDA @   2,ERTBL ;ERROR TABLE
        ISZ     ERTBL   ;INC THE POINTER
        INC     0,0     ;INC THE COUNT
        COM #   2,2,SNR ;SKIP IF NOT EOT
        JMP     .+3
        SUB #   2,3,SZR ;SKIP IF ERROR FOUND
        JMP     .-6     ;LOOP BACK
        JMP     .+2
SYSER:  ADC     0,0     ;SET FLAG -1
        STA     0,ERCOD ;HOLD ERROR COUNT
L5:     NIOC    MUX             ;RESET MUX
        NIOC    CLK     ;RESET CLOCK
        NIOC    DKP     ;RESET DISK PACK
        INTDS           ;DISABLE INTERRUPTS
        SUB     0,0     ;CLEAR AC0
        MSKO    0       ;CLEAR ALL INTERRUPT MASKS
        LDA     0,INTSV ;RETRIEVE RDOS INTERRUPT VECTOR
        STA     0,1
        INTEN           ;ENABLE INTERRUPTS
        COM #   1,1,SZR ;SKIP IF ERROR
        JMP @   OVRTN   ;NORMAL RETURN TO ROOT BINARY
        JMP @   ERRTN   ;ERROR RETURN

PH2CD:  1000            ;PHASE 2 ERROR CODE
ERTBL:  .+1             ;ERROR TABLE POINTER
```

```
        ERR1            ;DISK ERROR
        ERR2            ;OVERFLOW ERROR
        ERR3            ;MUX ERROR
        ERR4            ;OVERRUN ERROR
        -1              ;END OF TABLE

PNTRQ:  CTLTB           ;OUTPUT LIST POINTER
CTLOF:  BDACK           ;CONTROL LIST ENTRY SIZE

RTCCT:  0               ;RTC COUNT
SAVE:   0               ;INTERRUPT SAVE AREA
        0
        0
ENDFG:  0               ;END OF ACQUISITION/CONTROL FLAG
MUXFG:  0               ;MUX FLAG
DKPFG:  0               ;DISK FLAG
HLDFG:  0               ;HOLDOFF FLAG
INTSR:  SKPDN   CLK     ;SKIP IF RTC INTERRUPT
        JMP     .+5
        NIOS    CLK     ;RESTART RTC
        ISZ     RTCCT   ;INC COUNT
        INTEN
        JMP @   0       ;RETURN TO INTERRUPTED PROGRAM
        STA     0,SAVE  ;HOLD AC0, AC1
        STA     1,SAVE+1
        MOVL    0,0     ;SHIFT CARRY AND HOLD
        STA     0,SAVE+2
INTBK:  SKPDZ   MUX     ;SKIP IF NOT MUX INTERRUPT
        JMP     MUXIN
        DIAC    0,DKP   ;DKP STATUS, CLEAR FLAGS
        MOVR #  0,0,SZC ;SKIP IF NO DKP ERROR
        JSR @   ERR01
ERR1:   ADC     1,1     ;FORCE -1
        MOVL #  0,0,SNC ;SKIP IF WRITE INTERRUPT
        JMP     SKINT
        DSZ     BLKCT   ;DEC DATA BLOCK COUNT
        JMP     .+1
        LDA     0,ENDFG ;END OF ACQUISITION/CONTROL FLAG
        MOV #   0,0,SZR ;SKIP IF NOT DONE
        JMP     L4
        LDA     0,BLKCT ;DATA BLOCK COUNT
        MOVR #  0,0,SNC ;SKIP IF EVEN SURFACE COMPLETE
        JMP     DSKSK   ;SEEK NEW CYLINDER
SKINT:  INC     1,1     ;FORCE 0
        LDA     0,DKPFG ;DISK PACK FLAG
        STA     1,DKPFG ;UPDATE FLAG
        MOVZR # 0,0,SZR ;SKIP IF DEFERRED WRITE REQUIRED
        JMP     INTRL   ;RELEASE THE INTERRUPT
        LDA     0,MUXFG ;MUX FLAG
        JMP     WRITE
DSKSK:  MOV #   0,0,SNR ;SKIP IF NO OVERFLOW
        JSR @   ERR01
ERR2:   STA     1,DKPFG ;UPDATE FLAG
        ISZ     SEEKC   ;INC CYL # IN SEEK COMMAND
        ISZ     WRITC   ;INC CYL # IN WRITE COMMAND
        LDA     0,SEEKC
        DOAP    0,DKP   ;OUTPUT THE SEEK COMMAND
```

APPENDIX A

```
        JMP     INTRL   ;RELEASE THE INTERRUPT
MJXHF:  ISZ     HLDCT   ;INC THE HOLDOFF COUNT
INTRL:  INTA    0       ;INTERRUPT ACK.
        MOV #   0,0,SZR ;SKIP IF NO INTERRUPTS PENDING
        JMP     .+7
        LDA     0,SAVE+2
        MOVR    0,0     ;RETRIEVE CARRY
        LDA     1,SAVE+1
        LDA     0,SAVE  ;RETRIEVE ACI AND ACO
        INTEN
        JMP @   0       ;RETURN TO INTERRUPTED PROGRAM
        SKPDZ   CLK     ;SKIP IF NO CLOCK INTERRUPT
        JMP     .-7
        JMP     INTBK   ;LOOP BACK

MJXIN:  DIAS    0,MUX   ;RESTART MUX, GET STATUS
        LDA     1,MUXFG ;OLD STATUS
        STA     0,MUXFG ;UPDATE STUTUS
        SUB #   0,1,SNR ;SKIP IF NO MUX ERROR
        JSR @   ERR01
ERR3:   LDA     1,HLDFG ;HOLDOFF FLAG
        MOV #   1,1,SNR ;SKIP IF NO HOLDOFF IN EFFECT
        JMP     MJXHF
        LDA     1,DKPFG ;DISK FLAG
        MOV #   1,1,SZR ;SKIP IF DISK IDLE
        JMP     WRTDF   ;DEFER WRITING
WRITE:  STA     0,DKPFG ;SET DISK FLAG
        LDA     1,BUFS1 ;1ST BUFFER POINTER
        MOVL #  0,0,SNC ;SKIP IF 1ST BUFFER REQUIRED
        LDA     1,BUFS2 ;ELSE GET 2ND BUFFER POINTER
        DOB     1,DKP   ;OUTPUT THE POINTER
        LDA     1,SURFE ;EVEN SURFACE
        MOVL #  0,0,SNC ;SKIP IF EVEN SURFACE REQUIRES
        LDA     1,SURFO ;ELSE GET ODD SURFACE
        DOC     1,DKP
        LDA     1,WRITC ;WRITE COMMAND
        DOAS    1,DKP   ;OUTPUT COMMAND, START WRITE
        JMP     INTRL   ;RELEASE THE INTERRUPT
WRTDF:  MOVZR # 1,1,SZR ;SKIP IF DEFER IN EFFECT
        SUB #   0,1,SNR ;SKIP IF NO OVERRUN
        JSR @   ERR01
ERR4:   SUBZL   1,1     ;SET FLAG +1 (WRITE PENDING)
        STA     1,DKPFG ;DISK FLAG
        JMP     INTRL   ;RELEASE THE INTERRUPT

ERR01:  ERROR           ;ERROR ROUTINE

SEEKC:  175000+BDACD    ;INITIAL SEEK COMMAND
WRITC:  174400+BDACD    ;INITIAL WRITE COMMAND
SURFE:  000004          ;EVEN SURFACE
SURFO:  000404          ;ODD SURFACE

BUFS2:  .+2+BDACM       ;2ND BUFFER POINTER
BUFS1:  .+1             ;1ST BUFFER POINTER
        .BLK    BDACM*2 ;START OF DATA BUFFERS
        0

        .END
```

A-4. Overlay Module No. 3--PHAS3.SR

```
NAME BLOCK   NAME=  PHAS3.SR


TIME BLOCK


        .TITL   PHAS3   ;JCI    24 MAR 76

        .TXTM   0

        .ENT    OVST3

        .EXTD   OVRTN MAGFG ERCOD ERRTN HSMON

        .NREL

PNTRO:  HSBUF           ;HS BUFFER POINTER
CNTRO:  BDAC0           ;HS BUFFER SIZE
DMAOB:  MUXOB           ;MUX OUTPUT BUFFER POINTER
CNTRB:  BDAC2           ;OUTPUT BUFFER SIZE
NONE1:  NONE            ;NO HS BUFFER REQUIRED

OVST3:  SUB     0,0     ;FORCE 0
        STA @   0,PNTRO ;RESET HS OUTPUT BUFFER
        ISZ     PNTRO
        DSZ     CNTRO
        JMP     .-3
        ADC     0,0     ;FORCE -1
        STA @   0,DMAOB ;RESET MUX OUTPUT BUFFER
        ISZ     DMAOB
        DSZ     CNTRB
        JMP     .-3
        LDA     1,HSMON ;HS MONITOR FLAG
        MOV #   1,1,SNR ;SKIP IF ACTIVE
        JMP @   NONE1   ;WRAPUP

        SKPBZ   TTO     ;WAIT FOR TTO IDLE
        JMP     .-1
        NOP
        NOP
        SKPBZ   TTO
        JMP     .-5
        SKPDZ   TTO
        JMP     .-7
        LDA     1,SEEKG ;DISK SEEKING MASK
        DIA     0,DKP   ;DISK STATUS
        AND #   0,1,SZR ;SKIP IF NOT SEEKING
        JMP     .-2
        SKPBZ   DKP     ;WAIT FOR DISK IDLE
        JMP     .-1
        NOP
        NOP
        SKPBZ   DKP
        JMP     .-5
        SKPDZ   DKP
```

APPENDIX A

```
        JMP     .-7
        INTDS           ;DISABLE INTERRUPTS
        LDA     0,1     ;RDOS INT VECTOR
        LDA     1,INTSV ;BDACS INT VECTOR
        STA     0,INTSV ;SAVE RDOS VECTOR
        STA     1,1     ;ENABLE BDACS INT SYSTEM
        LDA     0,INTMK ;BDACS INT MASK
        MSKO    0       ;MASK OUT
        LDA     0,DMUXA ;DMUX I/O START ADDR. AND COUNT
        DOA     0,MUX
        LDA     0,DMUXB ;DCH INPUT START ADDR.
        DOB     0,MUX
        LDA     0,DMUXC ;HIGH SPEED WORD MASK
        DOC     0,MUX
        LDA     0,DMUXD ;DCH INPUT WORD COUNT
        DOCS    0,MUX   ;SET BUSY, RESET DONE

        LDA     2,TTYPT ;MSG POINTER
T1:     LDA     0,0,2   ;MESSAGE WORD
        INCO    2,2     ;INC POINTER
T2:     LDA     1,BTMSK ;BYTE MASK
        AND     0,1,SNR ;MASK THE BYTE, SKIP IF NOT NUL
        JMP     JUMP1
        DOAS    1,TTO   ;OUTPUT A CHARACTER
        SKPDN   TTO     ;SKIP WHEN DONE
        JMP     .-1
        MOVCS   0,0,SZC ;SWAP THE BYTES, SKIP IF VALID
        JMP     T1      ;GET NEXT WORD
        JMP     T2      ;OUTPUT NEXT CHARACTER

TIMER:  04              ;DELAY COUNT
BTMSK:  377             ;BYTE MASK
DMUXA:  000060          ;MUX START ADDRESS & COUNT
DMUXB:  INPUT-1         ;INPUT DMA BUFFER
DMUXC:  140144          ;HIGH RATE WORD
DMUXD:  077776          ;-WORD COUNT, AND 0 IN 1BO
DMUXE:  INPUT-2         ;INPUT DMA BUFFER (RUNNING)
DMUXF:  077775          ;-WORD COUNT (RUNNING)
DMUXG:  147777          ;LOW RATE WORD
SEEKG:  002000          ;DKP SEEK IN PROGRESS MASK
INTMK:  -1-1#MUX        ;BDACS INTERRUPT MASK
INTSV:  INTSR           ;BDACS INTERRUPT SERVICE ROUTINE
TTYPT:  .+1             ;POINTER TO TEXT
        .TXT    /<15><12>HS BUFFER RETRIEVAL<12><15><12>/

JUMP1:  NIOC    TTO     ;CLEAR TTO
        LDA     0,CNTLH
        STA     0,CNTLP
        LDA     0,DMUXE ;DCH INPUT START ADDR. (RUNNING)
        LDA     1,DMUXF ;DCH INPUT WORD COUNT (RUNNING)
        NIOP    MUX     ;START THE MULTIPLEXER
        DOB     0,MUX   ;OUTPUT RUNNING VALUES
        DOC     1,MUX

INTRL:  INTEN           ;ENABLE INTERRUPTS
        JMP     .       ;LOOP HERE TILL DONE
        JMP     .-1
```

```
MUXFG:  0                       ;MUX STATUS FLAG
CNTLL:  137777                  ;CONTROL WORD (LOW)
CNTLH:  037777                  ;CONTROL WORD (HIGH)
CNTLP:  MUXOB+BDAC2-1           ;CONTROL WORD POINTER
ERR01:  ERROR                   ;ERROR ROUTINE

        0
        0
INPUT:  0                       ;MUX INPUT BUFFER
        0
        0
INTSR:  DIAS    0,MUX           ;MUX STATUS
        LDA     1,MUXFG         ;PREVIOUS STATUS
        STA     0,MUXFG         ;UPDATE THE STATUS
        SUB #   0,1,SNR         ;SKIP IF NO ERROR
        JSR @   ERR01           ;ERROR ROUTINE
ERR1:   LDA     1,CNTLL         ;CONTROL WORD MASK (LOW SIGNAL)
        MOVL #  0,0,SNC         ;SKIP IF LOW REQUIRED
        LDA     1,CNTLH         ;CONTROL WORD MASK (HIGH SIGNAL)
        STA @   1,CNTLP         ;CONTROL WORD POINTER
        LDA     1,DMUXG         ;LOW RATE WORD
        MOVL #  0,0,SNC         ;SKIP IF LOW RATE REQUIRED
        LDA     1,DMUXC         ;HIGH RATE WORD
        DOC     1,MUX
        MOVL #  0,0,SZC         ;SKIP IF VALID DATA
        JMP     INTRL
        LDA     1,INPUT         ;DATA WORD
        MOVS    1,1             ;SWAP THE BYTES
        STA @   1,BFRPT         ;STORE THE WORD
        ISZ     BFRPT           ;INC THE POINTER
        DSZ     BFRCT           ;DEC THE COUNT, SKIP WHEN DONE
        JMP     INTRL           ;RELEASE THE INTERRUPT

        SUB     1,1,SKP         ;SET FLAG 0
ERROR:  ADC     1,1,SKP         ;SET FLAG -1
        JMP     DONE
        LDA     0,PHSCD         ;PHASE 3 ERROR CODE
        LDA @   2,ERTBL         ;ERROR TABLE
        ISZ     ERTBL           ;INC THE POINTER
        INC     0,0             ;INC THE COUNT
        COM #   2,2,SNR         ;SKIP IF NOT EOT
        JMP     .+3
        SUB #   2,3,SZR         ;SKIP IF ERROR FOUND
        JMP     .-6             ;LOOP BACK
        JMP     .+2
SYSER:  ADC     0,0             ;SYSTEM ERROR CODE
        STA     0,ERCOD         ;HOLD ERROR COUNT
DONE:   NIOC    MUX             ;RESET MUX
        INTDS                   ;DISABLE INTERRUPTS
        SUB     0,0             ;CLEAR AC0
        MSKO    C               ;CLEARMALL INTERRUPT MASKS
        LDA     0,INTSV         ;RETRIEVE RDOS INTERRUPT VECTOR
        STA     0,1
        INTEN                   ;ENABLE INTERRUPTS
NONE:   COM #   1,1,SNR         ;SKIP IF NO ERROR
        JMP @   ERRTN
        JSR @   CRRF1           ;CREATE
```

APPENDIX A

```
                NAMEB   ;'TEMPB.TM'
        JSR @   OPFL1   ;OPEN
                NAMEB   ;'TEMPB.TM'
                03      ;ON CH #3
        JSR @   TMPO1   ;OUTPUT HS BUFFER
        JSR @   CLFL1   ;CLOSE
                03      ;CH #3
        LDA     1,MAGFG ;MAG TAPE FLAG
        MOV #   1,1,SNR ;SKIP IF MAG TAPE
        JMP @   OVRTN   ;RETURN TO ROOT BINARY
        JSR @   OPFL1   ;OPEN
                NAMEB   ;'TEMPB.TM'
                04      ;ON CH #4
        JSR @   OPMT1   ;OPEN MAG TAPE FOR FF
                NAMEA   ;'MTO:5'
                03      ;ON CH #3
        SUB     0,0     ;CLEAR AC0
        STA     0,BLKND ;BLOCK #
        JSR @   RDBL1   ;READ A DISK BLOCK
BLKND:          0       ;BLOCK #
                .+4     ;EOF RETURN
        JSR @   MTAW1   ;WRITE BLOCK TO MTA
        ISZ     BLKND   ;INC BLOCK #
        JMP     .-5
        JSR @   MTAE1   ;WRITE EOF
        JSR @   MTAS1   ;SPACE REVERSE
        JSR @   CLFL1   ;CLOSE
                03      ;CH #3
        JSR @   CLFL1   ;CLOSE
                04      ;CH #4
        JMP @   OVRTN   ;RETURN TO ROOT BINARY

BFRST:  HSBUF+1         ;START OF HS BUFFER
BFRPT:  HSBUF           ;BUFFER POINTER
BFRCT:  BDAC0+1         ;BUFFER COUNT
BLKCT:  BDAC0/400       ;# BLOCKS IN HS BUFFER

CRRF1:  CRRFL           ;CREATE RANDOM FILE
OPFL1:  OPFLE           ;OPEN A CHANNEL
CLFL1:  CLFLE           ;CLOSE A CHANNEL
OPMT1:  OPMTA           ;OPEN MAG TAPE FOR FF
RDBL1:  RDBLK           ;READ A DISK BLOCK
TMPO1:  TMPOT           ;OUTPUT HS BUFFER TO DISK
MTAW1:  MTAWT           ;OUTPUT BLOCK TO MAG TAPE
MTAE1:  MTAEF           ;WRITE EOF ON MTA
MTAS1:  MTASR           ;SPACE REVERSE 1 RECORD

PH3CD:  1400            ;PHASE 3 ERROR CODE
ERTBL:  .+1             ;ERROR TABLE POINTER
        ERR1            ;MUX ERROR
        -1              ;END OF TABLE

C11:    11
C12:    12
CRRTN:  0               ;RETURN ADDRS
CRRFL:  LDA     0,0,3   ;NAME POINTER
        INC     3,3
```

```
        STA     3,CRRTN ;STORE RETURN
CRRFA:  STA     3,USP   ;STORE RETURN
        .SYSTM
        .CRAND          ;CREATE RANDOM FILE
        JMP     .+2
        JMP     0,3     ;NORMAL RETURN
        LDA     1,C11
        SUB #   1,2,SZR ;SKIP IF FILE EXISTS
        JSR @   SYSE1   ;ERROR
        JSR     DELEA   ;DELETE THE FILE
        JSR     CRRFA   ;TRY AGAIN
        JMP @   CRRTN   ;RETURN

DELET:  LDA     0,0,3   ;NAME POINTER
        INC     3,3
DELEA:  STA     3,USP   ;STORE RETURN
        .SYSTM
        .DELET          ;DELETE THE FILE
        JMP     .+2
        JMP     0,3     ;NORMAL RETURN
        LDA     1,C12
        SUB #   1,2,SZR ;SKIP IF NO FILE EXISTS
        JSR @   SYSE1   ;ERROR
        JMP     0,3     ;NORMAL RETURN

OPMTA:  ADC     1,1,SKP ;SET FLAG -1
OPFLE:  SUB     1,1     ;SET FLAG 0
        LDA     0,0,3   ;NAME POINTER
        LDA     2,1,3   ;CHANNEL #
        STA     3,USP   ;HOLD RETURN
        MOV #   1,1,SZR ;SKIP IF REGULAR OPEN
        JMP     .+5
        .SYSTM
        .OPEN   77      ;OPEN THE CHANNEL
        JSR @   SYSE1   ;ERROR
        JMP     2,3     ;NORMAL RETURN
        INC     1,1     ;DEFAULT CHARACTERISTICS
        .SYSTM
        .MTOPD  77      ;OPEN MAG TAPE FOR FF
        JMP @   SYSE1   ;ERROR
        JMP     2,3     ;NORMAL RETURN

SYSE1   :SYSER          ;SYSTEM ERROR

EOFMK:  000400          ;EOF ERROR MASK
SRCMD:  040001          ;SPACE REVERSE COMMAND
EFCMD:  060000          ;EOF WRITE COMMAND
WTCMD:  050400          ;WRITE COMMAND
MTASR:  LDA     1,SRCMD         ;SPACE REVERSE COMMAND
        JMP     .+2
MTAEF:  LDA     1,EFCMD ;EOF WRITE COMMAND
        JMP     .+3
MTAWT:  LDA     0,BFRST ;START OF HS BUFFER
        LDA     1,WTCMD ;WRITE COMMAND
        STA     3,USP   ;HOLD RETURN
        .SYSTM
        .MTDIO  03      ;MAG TAPE FF I/O
```

APPENDIX A

```
        JMP     .+2
        JMP     0,3         ;NORMAL RETURN
        LDA     1,EOFMK     ;EOF ERROR MASK
        AND #   1,2,SNR     ;SKIP IF EOF
        JSR @   SYSE1       ;ERROR
        JMP     0,3         ;NORMAL RETURN

TMPCT:  LDA     0,BFRST     ;START OF HS BUFFER
        SUB     1,1         ;RELATIVE BLOCK ADDRS 0
        LDA     2,BLKCT     ;# BLOCKS
        MOVS    2,2
        STA     3,USP       ;STORE RETURN
        .SYSTM
        .WRB    03          ;WRITE THE HS BUFFER TO DISK
        JSR @   SYSE1       ;ERROR
        JMP     0,3         ;NORMAL RETURN

RDBAK:  0                   ;RETURN ADDRS
RDBLK:  LDA     1,0,3       ;BLOCK #
        LDA     0,1,3       ;EOF RETURN ADDRS
        STA     0,RDBAK     ;STORE ADDRS
        LDA     0,BLKPT     ;BLOCK BUFFER POINTER
        LDA     2,BLKSZ     ;BLOCK SIZE
        STA     3,USP       ;STORE RETURN
        .SYSTM
        .RDB    04          ;READ A DISK BLOCK
        JMP     .+2         ;ERROR
        JMP     2,3         ;NORMAL RETURN
        LDA     1,EOFCD     ;EOF CODE
        SUB #   1,2,SZR     ;SKIP IF EOF
        JSR @   SYSE1       ;SYSTEM ERROR
        JMP @   RDBAK       ;EOF RETURN

BLKPT:  HSBUF               ;BLOCK BUFFER POINTER
BLKSZ:  000400              ;BLOCK COUNT 1
EOFCD:  06                  ;EOF ERROR CODE

CLOSE:  LDA     2,0,3       ;CH #
        STA     3,USP       ;HOLD RETURN
        .SYSTM
        .CLOS   77          ;CLOSE THE CHANNEL
        JMP @   SYSE1
        JMP     1,3         ;NORMAL RETURN

        .TXTM   1

NAMEA=  .+2
        .TXT    /MTO:5/

NAMEB=  .+2
        .TXT    /TEMPB.TM/

HSBUF=  .
        .BLK    BDACC+1 ;HS OUTPUT BUFFER
        0

        .END
```

A-5. Overlay Module No. 4--PHAS4.SR

```
NAME BLOCK   NAME=  PHAS4.SR

TIME BLOCK

        .TITL   PHAS4   ;JCI     10 MAR 76

        .FXTM   1

        .ENT    OVST4

        .EXTD   OVRTN ERRTN ERCOD MAGFG SMPRT
        .EXTD   BLKCT HLDCT LSMON

        .EXTN   MSKTB LSBTB

        .NREL

BUFSZ:  BDAC9           ;REDUCED DATA BUFFER SIZE
BUFPT:  MTABF           ;BUFFER POINTER
CRRF1:  CRRFL           ;CREATE RANDOM FILE
OPFL1:  OPFLE           ;OPEN A FILE (NORMAL)
OPMT1:  OPMTA           ;OPEN FF MAG TAPE FILE
DONE1:  DONE            ;WRAPUP

OVST4:  SUB     0,0     ;CLEAR AC0
        STA @   0,BUFPT ;RESET REDUCED DATA BUFFER
        ISZ     BUFPT
        DSZ     BUFSZ
        JMP     .-3
        JSR @   OPFL1   ;OPEN
                NAMEC   ;'BDACS.DA'
                04      ;ON CH #4
        LDA     1,MAGFG ;MAG TAPE FLAG
        MOV #   1,1,SZR ;SKIP IF NO MAG TAPE
        JMP     .+7
        JSR @   CRRF1   ;CREATE A RANDOM FILE
                NAMEB   ;'TEMPC.TM'
        JSR @   OPFL1   ;OPEN
                NAMEB   ;'TEMPC.TM'
                03      ;ON CH#3
        JMP     .+4
        JSR @   OPMT1   ;FF OPEN
                NAMEA   ;'MT0:3'
                03      ;ON CH#3
        LDA     1,BLKST ;DATA BLOCK START COUNT
        LDA     0,BLKCT ;CURRENT COUNT
        SUB     1,0     ;FORM NEG DIFF
        MOV #   0,0,SNR ;SKIP IF DATA STORED
        JMP @   DONE1
        STA     0,BLKCT ;SET COUNT
        LDA     1,SMPRT ;SAMPLE RATE
        SUB     0,0     ;CLEAR AC0, AC2, AC3
        MOV     0,2
```

APPENDIX A

```
        MOV     0,3
        ADDZ    1,3,SZC ;DETERMINE HOLDOFF TIME PER FILL
        INC     2,2
        DSZ     CNTR0   ;SKIP WHEN DONE
        JMP     .-3
        LDA     1,HLDCT ;HOLDOFF COUNT
        MOV #   1,1,SNR ;SKIP IF HOLDOFF
        JMP     LL0
        STA     1,CNTR0 ;HOLD COUNT
        MOV     0,1     ;CLEAR AC1
        ADDZ    3,1,SZC ;DETERMINE TOTAL HOLDOFF TIME
        INCZ    0,0,SNC ;SKIP IF OVERFLOW
        ADDZ    2,0,SZC ;SKIP IF NO OVERFLOW
        JSR @   ERR01   ;ERROR ROUTINE
ERR1:   DSZ     CNTR0   ;SKIP WHEN DONE
        JMP     .-5
        STA     0,TIMER ;STORE THE TIME
        STA     1,TIMER+1
        LDA     0,HLDCT ;GET HOLDOFF COUNT
        STA     0,CNTR0 ;RESET THE COUNT
        SUB     0,0     ;CLEAR AC0
        LDA     1,HLDOV ;ROUND ROBIN OVERFLOW PER FILL
        LDA     2,RRCNT ;ROUND ROBIN COUNT
        ADD     1,0     ;ADD OVERFLOW
        SUBZ #  2,0,SZC ;SKIP IF AC2>AC0
        SUB     2,0     ;ELSE RESET THE OVERFLOW
        DSZ     CNTR0   ;SKIP WHEN DONE
        JMP     .-4
        MOV #   0,0,SZR ;SKIP IF NO NET OFFSET
        STA     0,PRADD ;UPDATE PREVIOUS ADDRESS

LL0:    LDA     2,LSMON ;LS BUFFER MONITOR POINT
        ADCZ    0,0     ;FORCE -1,CLEAR CARRY
        NEG     2,2,SZR ;SKIP IF NO LS
        MOVL    0,0,SKP ;FORM THE LS MASK
        JMP     .+3
        INC     2,2,SZR ;SKIP WHEN DONE
        JMP     .-3
        STA     0,LSBMK ;STORE THE MASK
        READS   0       ;READ SWITCHES
        ADDZL # 0,0,SNC ;SKIP IF NO ERROR RECOVERY
        JMP     LL1
LL:     JSR @   GTBW1   ;GET A LS WORD
        LDA     2,PRADD ;GET PREVIOUS LS OFFSET
        LDA     1,RRCNT ;MAX COUNT
        ADCZ #  2,1,SNC ;SKIP IF AC2<AC1
        SUB     1,2     ;RESET THE OFFSET
        INC     2,2     ;INC THE OFFSET
        STA     2,PRADD ;RESTORE THE CURRENT OFFSET
        JSR @   RDBW1   ;REDUCE LS WORD
        JSR @   GTBW1   ;GET A HS WORD
        SUB     2,2     ;0 OFFSET
        JSR @   RDBW1   ;REDUCE HS WORD
        JSR     INTMR   ;INC THE TIMER
        JMP     LO      ;LOOP BACK
```

```
HLDDV:  BDAC7+BDAC8/BDAC7*BDAC7-BDAC8
PRADD:  BDAC7           ;PREVIOUS LS ADDRS
RRCNT:  BDAC7           ;# OF LS WORDS
BLKST:  BDACE*2         ;MAX DATA BLOCK COUNT
CNTRO:  BDAC8           ;# OF HS/LS PAIRS PER BUFFER
LSMSK:  160000          ;LS ADDRS MASK
WDSTR:  0               ;DATA WORD STORAGE
HSPNT:  MSKTB           ;REPORTED POINTS MASK TABLE

GTBW1:  GTBWD           ;GET A BUFFER WORD
RDBW1:  RDBWD           ;REDUCE A BUFFER WORD
ERRO1:  ERROR           ;ERROR ROUTINE

TIMER:  0               ;CURRENT TIME
        0

INTMR:  LDA     0,SMPRT ;SAMPLE RATE
        LDA     1,TIMER ;CURRENT TIME
        LDA     2,TIMER+1
        ADDZ    0,2,SZC ;UPDATE THE TIMER
        INCZ    1,1,SNC ;SKIP IF OVERFLOW
        JMP     .+2
        JSR @   ERRO1   ;ERROR ROUTINE
ERR2:   STA     1,TIMER ;RESTORE TIMER
        STA     2,TIMER+1
        LDA     0,LSMON ;LS BUFFER MONITOR POINT #
        MOV #   0,0,SNR ;SKIP IF BUFFER IN USE
        JMP     0,3     ;RETURN
        ISZ     LSPNT   ;INC LS POINT COUNT
        LDA     0,LSWMK ;LS WORD MASK
        MOVZL   0,0,SZR ;SHIFT, SKIP IF OVERFLOW
        JMP     .+3
        ISZ     LSOFF   ;INC OFFSET COUNT
        SUBZL   0,0     ;RESET MASK
        STA     0,LSWMK ;RESTORE THE MASK
        LDA     0,LSOFF ;GET OFFSET
        LDA     1,LSMAX ;MAX OFFSET
        SUBZ #  0,1,SZC ;SKIP IF AC0>AC1
        JMP     .+5
        SUB     0,0     ;CLEAR AC0
        STA     0,LSOFF ;RESET THE OFFSET
        LDA     0,LSFST ;1ST LS POINT
        STA     0,LSPNT ;RESET THE POINT COUNT
        JMP     0,3     ;RETURN

LL1:    SUBZL   0,0     ;+1 INITIAL HS COUNT
        STA     0,CNTRO ;HS COUNT
L1:     SUB     0,0     ;CLEAR AC0
        STA     0,SEQFG ;RESET ERROR FLAG
        JSR @   GTBW1   ;GET A BUFFER WORD
        MOVL    0,0,SNC ;SKIP IF HS WORD
        JMP     L2
        MOVZR   0,0     ;RESET HS BIT
        SUB     2,2     ;0 OFFSET
        JSR @   RDBW1   ;REDUCE HS WORD
        ISZ     CNTRO   ;INC HS COUNT
        JSR     INTMR   ;INC TIMER
```

APPENDIX A

```
        JMP     L1          ;LOOP BACK
L2:     LDA     1,LSMSK     ;LS MASK
        AND     0,1         ;MASK THE ADDRS BITS
        SUB     1,C         ;FORM DATA WORD
        MOVZR   0,0
        STA     0,WDSTR     ;STORE THE DATA
        ADDZL   1,1         ;SHIFT ADDRS BITS
        MOVL    1,1
        MOVL    1,1
        LDA     2,PRADD     ;PREVIOUS LS ADDRS
        STA     1,PRADD     ;STORE CURRENT LS ADDRS
        LDA     0,RRCNT     ;ROUND ROBIN COUNT
        ADCZ #  2,1,SNC     ;SKIP IF AC2<AC1
        ADD     0,1         ;ADD MODULUS
        SUB     2,1         ;FORM ADDRS DIFF
        ADC     2,2         ;FORCE -1
        LDA     0,CNTRO     ;HS COUNT
        ADD #   2,1,SNR     ;SKIP IF DIFF NOT +1
        ADD #   ,0,SZR      ;SKIP IF HS COUNT +1
        JMP     L4
        INC     2,2         ;CLEAR AC2
        STA     2,CNTRO     ;RESET HS COUNT
L3:     LDA     0,WDSTR     ;RETRIEVE THE DATA
        LDA     2,PRADD     ;LS OFFSET
        JSR @   ROBW1       ;REDUCE LS WORD
        JMP     L1          ;LOOP BACK
L4:     SUBZ    0,1,SNC     ;FORM DIFF, SKIP IF ACO<=AC1
        JSR @   ERRO1       ;ERROR ROUTINE
ERR3:   STA     1,CNTRO     ;FORCE COUNT TO DIFF
        ISZ     SEQER       ;INC ERROR COUNT
        ISZ     SEQFG       ;SET ERROR FLAG
        LDA     0,SEQER     ;ERROR COUNT
        LDA     2,SEQMX     ;MAX COUNT
        SUBZ #  2,0,SZC     ;SKIP IF AC2>ACO
        JSR @   ERRO1       ;ERROR ROUTINE
ERR4:   MOV #   1,1,SNR     ;SKIP IF NONZERO COUNT
        JMP     L3
        JSR     INTMR       ;INC TIMER
        DSZ     CNTRO       ;SKIP WHEN DONE
        JMP     .-2
        JMP     L3

SEQMX:  BDACN               ;MAX # SEQUENCE ERRORS
SEQER:  0                   ;SEQUENCE ERRORS
SEQMK:  000400              ;SEQUENCE ERROR MASK
SEQFG:  0                   ;SEQUENCE ERROR FLAG

LSBMK:  177777              ;LS BUFFER MASK
LSWMK:  000001              ;LS WORD MASK
LSFST:  040000+BDACO        ;FIRST LS POINT #
LSPNT:  040000+BDACO        ;CURRENT LS POINT #
LSMAX:  BDAC3-1             ;MAX LS OFFSET
LSOFF:  0                   ;CURRENT LS OFFSET
LSTPT:  LSBTB               ;START OF LS MASK TABLE
LSRTN:  0                   ;RETURN ADDRS
```

```
LSBRD:  STA     3,LSRTN ;STORE RETURN
        LDA     3,LSTPT ;START OF TABLE
        LDA     0,LSOFF ;OFFSET
        ADD     0,3     ;THE POINTER
        LDA     1,0,3   ;REPORTED POINTS MASK
        LDA     0,LSWMK ;WORD MASK
        AND #   0,1,SNR ;SKIP IF REPORTED
        JMP     LS1
        LDA     1,BDAC3,3       ;PREVIOUS VALUE
        MOVL    2,2     ;MOV CURRENT STATE TO CARRY
        MOV     1,2,SZC ;MOV PREV. VALUE, SKIP IF 0 CURRENT STATE
        COM     2,2     ;COM PREVIOUS VALUE
        AND #   0,2,SNR ;SKIP IF CHANGE OCCURRED
        JMP     LS1
        COM     0,0     ;COM WORD MASK
        AND     0,1,SZC ;TURN OFF BIT, SKIP IF TO TURN OFF
        ADC     0,1     ;TURN ON
        STA     1,BDAC3,3       ;RESTORE UPDATED MASK
        LDA     1,LSPNT ;LS BUFFER POINT #
        SUBCR   2,2,SKP ;RETRIEVE CURRENT STATE BIT
LS1:    SUB     1,1     ;SET FLAG 0
        JMP @   LSRTN   ;RETURN

RDBWD:  STA     3,RDRTN ;RETURN ADDRS
        LDA     3,HSPNT ;MASK TABLE POINTER
        ADD     2,3     ;ADD IN OFFSET
        LDA     1,0,3   ;REPORTED POINTS MASK
        AND     1,0     ;MASK THE WORD
        LDA     1,BDAC1,3       ;PREVIOUS VALUE
        STA     0,BDAC1,3       ;UPDATE PREVIOUS VALUE
        ADDZL   2,2     ;FORM POINT # (MULT BY 16)
        ADDZL   2,2
        STA     2,STNBR ;HOLD THE POINT #
        COM     0,2     ;COM PRESENT VALUES
        COM     1,3     ;COM PAST VALUES
        AND     2,3     ;FORM SAME 0'S MASK
        AND     0,1     ;FORM SAME 1'S MASK
        ADD     3,1     ;MASK OF NON CHANGED POINTS
        LDA     3,LSMON ;LS BUFFER MONITOR POINT
        LDA     2,STNBR ;PRESENT #
        MOV #   3,3,SZR ;SKIP IF NO MONITOR BUFFER
        MOV #   2,2,SZR ;SKIP IF HS WORD
        JMP     .+3
        LDA     2,LSBMK ;LS BUFFER MASK
        AND     2,1     ;SET THE LS BIT
        COM     1,1     ;FORM CHANGES MASK
LP1:    MOV #   1,1,SNR ;SKIP IF MORE CHANGES
        JMP @   RDRTN   ;RETURN
        ISZ     STNBR   ;INC POINT #
        MOVR    0,0     ;MOV PRESENT VALUE TO CARRY
        SUBCR   2,2     ;MOV VALUE TO IBO
        MOVZR   1,1,SNC ;SKIP IF CHANGE
        JMP     LP1
        STA     0,HLDPR ;HOLD THE REMAINING VALUES
        STA     1,HLDDF ;HOLD THE DIFF MASK
        LDA     1,STNBR ;RETRIEVE POINT #
        LDA     0,LSMON ;LS BUFFER MONITOR POINT
```

APPENDIX A

```
        SUB #   0,1,SNR ;SKIP IF NOT LS POINT
        JSR     LSBRD   ;REDUCE THE LS POINT
        MOV #   1,1,SNR ;SKIP IF STORAGE REQUIRED
        JMP     LP2
        ADD     1,2     ;ADD IN POINT #
        SUB     1,1     ;CLEAR AC2
        LDA     3,SEQFG ;SEQUENCE ERROR FLAG
        MOV #   3,3,SZR ;SKIP IF NO ERROR
        LDA     1,SEQMK ;SEQUENCE ERROR MASK
        ADD     1,2     ;ADD IN ERROR MASK
        STA @   2,MTABP ;STORE STATE,POINT #, AND ERROR FLAG
        ISZ     MTABP   ;INC THE POINTER
        LDA     2,TIME  ;TIME POINTER
        LDA     1,0,2   ;MS TIME
        STA @   1,MTABP
        ISZ     MTABP
        LDA     1,1,2   ;LS TIME
        STA @   1,MTABP
        ISZ     MTABP
        DSZ     MTACT   ;SKIP WHEN BUFFER FULL
        JMP     LP2
        LDA     1,MAGFG ;MAG TAPE FLAG
        MOV #   1,1,SZR ;SKIP IF NO MAG TAPE
        JMP     .+3
        JSR @   TMPO1   ;WRITE BUFFER TO DISK FILE
        JMP     .+2
        JSR @   MTAO1   ;WRITE BUFFER TO MAG TAPE
        LDA     1,MTAST ;RESET INITIAL BUFFER POINTER
        STA     1,MTABP
        LDA     1,MTASZ ;RESET BUFFER COUNT
        STA     1,MTACT
LP2:    LDA     0,HLDPR ;RETRIEVE PRESENT VALUES
        LDA     1,HLDDF ;RETRIEVE DIFF MASK
        JMP     LP1     ;LOOP BACK

RDRTN:  0               ;RETURN ADDRS
STNBR:  0               ;POINT #
HLDPR:  0               ;PRESENT VALUES STORAGE
HLDDF:  0               ;DIFF MASK STORAGE
MTAST:  MTABF           ;MAG TAPE BUFFER START
MTABP:  MTABF           ;BUFFER POINTER
MTASZ:  400             ;REDUCED DATA BUFFER SIZE
MTACT:  400             ;BUFFER COUNT
TIME:   TIMER           ;CURRENT TIME POINTER
CM03:   -BDACL          ;SIZE OF REDUCED DATA BUFFER ENTRY (NEG)
C20:    20
CNTR1:  0               ;BIT COUNT

OPFL2:  OPFLE           ;OPEN A CHANNEL
CLFL1:  CLFLE           ;CLOSE A CHANNEL
TMPO1:  TMPOT           ;DISK OUTPUT ROUTINE
MTAO1:  MTAOT           ;MAG TAPE OUTPUT ROUTINE
MTAE1:  MTAEW           ;WRITE MAG TAPE EOF
MTAS1:  MTASR           ;SPACE REVERSE 1 RECORD

DONE:   SUB     0,0     ;CLEAR AC0
        LDA     2,CM03  ;ITEM ENTRY SIZE
```

```
        MOV     2,1
        STA @   0,MTABP ;NULL FILL REMAINING BUFFER SPACE
        ISZ     MTABP   ;INC POINTER
        INC     1,1,SZR ;ENTRY COMPLETE
        JMP     .-3
        DSZ     MTACT   ;SKIP WHEN DONE
        JMP     .-6
        LDA     1,MAGFG ;MAG TAPE FLAG
        MOV #   1,1,SZR ;SKIP IF NO MAG TAPE
        JMP     .+3
        JSR @   TMPO1   ;WRITE BUFFER TO DISK FILE
        JMP     .+4
        JSR @   MTAO1   ;WRITE BUFFER TO MAG TAPE
        JSR @   MTAE1   ;WRITE EOF
        JSR @   MTASI   ;SPACE REVERSE I RECORD
        JSR @   CLFL1   ;CLOSE
                03      ;CH #3
        JSR @   CLFL1   ;CLOSE
                04      ;CH #4

        JMP @   OVRTN   ;RETURN TO ROOT BINARY

SYSCD:  0               ;SYSTEM ERROR CODE
PH4CD:  2000            ;PHASE 4 ERROR CODE
ERTBL:  .+1             ;ERROR TABLE POINTER
        ERR1
        ERR2
        ERR3
        ERR4
        -1              ;END OF TABLE
ERROR:  INTEN           ;ENABLE INTERRUPTS
        LDA     0,PH4CD ;PHASE 4 ERROR CODE
        LDA @   2,ERTBL ;ERROR TABLE ENTRY
        ISZ     ERTBL   ;INC TABLE POINTER
        INC     0,0     ;INC COUNT
        COM #   2,2,SNR ;SKIP IF NOT EOT
        JMP     .+3
        SUB #   2,3,SZR ;SKIP IF ERROR FOUND
        JMP     .-6
        JSR @   CLFL1   ;CLOSE
                03      ;CH #3
        JSR @   CLFL1   ;CLOSE
                04      ;CH #4
        JMP     .+2
SYSER:  ADC     0,0     ;SYSTEM ERROR FLAG
        STA     0,ERCOD ;STORE IN ROOT BINARY
        JMP @   ERRTN   ;ERROR RETURN

GTBWD:  STA     3,GTRTN ;STORE RETURN ADDRS
        DSZ     BFRCT   ;DEC COUNT, SKIP WHEN EMPTY
        JMP     GTBUF
        LDA     0,BLKCT ;DATA BLOCK COUNT
        MOV #   0,0,SNR ;SKIP IF NOT DONE
        JMP @   DONE2
        INC     0,0     ;INC THE COUNT
        STA     0,BLKCT ;RESTORE THE COUNT
        LDA     1,BLOCK ;CURRENT DATA BLOCK
```

APPENDIX A

```
         LDA     0,BLKOF ;BLOCK OFFSET
         MOVS    0,2     ;SET BLOCK COUNT
         ADD     1,0     ;UPDATE CURRENT BLOCK
         STA     0,BLOCK
         LDA     0,BFRST ;START OF DATA BUFFER
         STA     0,BFRPT ;RESET THE BUFFER POINTER
         .SYSTM
         .RDB    04      ;READ THE DATA BLOCK INTO BUFFER
         JMP  @  SYSE1   ;SYSTEM ERROR
         LDA     0,BFRSZ ;BUFFER SIZE
         STA     0,BFRCT ;RESET THE COUNT
CTBUF:   LDA  @  0,BFRPT ;GET WORD FROM BUFFER
         MOVS    0,0     ;SWAP THE BYTES
         ISZ     BFRPT   ;INC THE POINTER
         READS   1       ;READ SWITCHES
         MOVR #  1,1,SNC ;SKIP IF 1B15
         JMP  @  GTRTN   ;RETURN
         JSR     DMPWD   ;DUMP THE WORD
         JMP  @  GTRTN   ;RETURN

BFRST:   BUFFR           ;START OF BUFFER
BFRPT:   BUFFR           ;BUFFER POINTER
BFRSZ:   BDACM           ;BUFFER SIZE
BFRCT:   01              ;BUFFER COUNT
BLOCK:   0               ;CURRENT DATA BLOCK
BLKOF:   14              ;BLOCK OFFSET (14 SECTORS)

TYPM1:   TYPMG           ;TYPE A MESSAGE
SYSE1:   SYSER           ;SYSTEM ERROR ROUTINE
DONE2:   DONE            ;WRAPUP

GTRTN:   0               ;RETURN ADDRS STORAGE
DPRTN:   0               ;RETURN ADDRS STORAGE
DPSTR:   0               ;DATA STORAGE
DPBPT:   DPMSG/2         ;WORD POINTER
CM10:    -10             ;WORD COUNT
C60:     60              ;<0>
PNTRO:   0               ;UTILITY POINTER

DMPWD:   STA     0,DPSTR ;HOLD DATA WORD
         STA     3,DPRTN ;STORE RETURN ADDRS
         LDA     3,CM10  ;SET WORD COUNT
         LDA     1,DPBPT ;BUFFER POINTER
         STA     1,PNTRO
K1:      LDA     1,C60   ;<0>
         MOV     1,2
         MOVL    0,0,SZC ;SKIP IF BIT 0
         INC     1,1     ;<1>
         MOVS    1,1     ;SWAP THE BYTE
         MOVL    0,0,SZC ;SKIP IF BIT 0
         INC     2,2     ;<1>
         ADD     2,1     ;ADD RH BYTE
         STA  2  1,PNTRO ;STORE THE BYTES
         ISZ     PNTRO   ;INC THE POINTER
         INC     3,3,SZR ;INC COUNT, SKIP WHEN DONE
         JMP     K1      ;LOOP BACK
         JSR  @  TYPM1   ;TYPE THE MESSAGE
```

```
                DPMSG   ;WORD DUMP
        LDA     0,DPSTR ;RETRIEVE DATA WORD
        JMP @   DPRTN   ;RETURN

CLFLE:  LDA     2,0,3   ;CH #
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .CLOS   77      ;CLOSE THE CHANNEL
        JMP @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

DPMTA:  ADC     1,1,SKP ;SET FLAG +1
DPFLE:  SUB     1,1     ;SET FLAG 0
        LDA     0,0,3   ;NAME POINTER
        LDA     2,1,3   ;CH#
        STA     3,USP   ;STORE RETURN
        MOV #   1,1,SZR ;SKIP IF REGULAR OPEN
        JMP     .+5
        .SYSTM
        .OPEN   77      ;OPEN THE CHANNEL
        JMP @   SYSE1   ;ERROR
        JMP     2,3     ;NORMAL RETURN
        INC     1,1     ;DEFAULT CHARACTERISTICS
        .SYSTM
        .MTOPD  77      ;OPEN THE CHANNEL
        JMP @   SYSE1   ;ERROR
        JMP     2,3     ;NORMAL RETURN

C11:    11
C12:    12
CRRTN:  0               ;RETURN ADDRS
CRRFL:  LDA     0,0,3   ;NAME POINTER
        INC     3,3
        STA     3,CRRTN ;STORE RETURN
CRRFA:  STA     3,USP   ;STORE RETURN
        .SYSTM
        .CRAND          ;CREATE RANDOM FILE
        JMP     .+2
        JMP     0,3     ;NORMAL RETURN
        LDA     1,C11
        SUB #   1,2,SZR ;SKIP IF FILE EXISTS
        JMP @   SYSE1
        JSR     DELEA   ;DELETE THE FILE
        JSR     CRRFA   ;TRY AGAIN
        JMP @   CRRTN   ;RETURN
DELET:  LDA     0,0,3   ;NAME POINTER
        INC     3,3
DELEA:  STA     3,USP   ;STORE RETURN
        .SYSTM
        .DELET          ;DELETE THE FILE
        JMP     .+2
        JMP     0,3     ;NORMAL RETURN
        LDA     1,C12
        SUB #   1,2,SZR ;SKIP IF NO FILE EXISTS
        JMP @   SYSE1   ;ERROR
        JMP     0,3     ;NORMAL RETURN
```

APPENDIX A

```
TYPMG:  LDA     0,0,3   ;BYTE POINTER
        SUBZL   2,2     ;FORCE CH #1
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .WRL    77      ;WRITE A LINE
        JMP @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

MTASR:  LDA     1,MTASC ;SPACE REVERSE COMMAND
        JMP     .+5
MTAEW:  LDA     1,MTAEC ;EOF WRITE COMMAND
        JMP     .+3
MTAOT:  LDA     0,MTAB1 ;REDUCED DATA OUTPUT BUFFER
        LDA     1,MTACM ;WRITE FF COMMAND
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .MTDIO  03      ;OUTPUT BUFFER TO MAG TAPE
        JMP     .+2
        JMP     0,3     ;NORMAL RETURN
        LDA     1,MTEOF ;EOF ERROR MASK
        AND #   1,2,SNR ;SKIP IF EOF
        JSR @   SYSE1   ;ERROR
        JMP     0,3     ;NORMAL RETURN

TMPOT:  LDA     0,MTAB1 ;REDUCED DATA OUTPUT BUFFER
        LDA     1,TMPBK ;CURRENT BLOCK #
        STA     3,USP   ;STORE RETURN
        MOV     1,3     ;HOLD THE BLOCK #
        LDA     2,TMPCT ;# WORDS IN BUFFER ENTRY
        ADD     2,3     ;UPDATE BLOCK #
        STA     3,TMPBK
        MOVS    2,2     ;SWAP BLOCK COUNT
        .SYSTM
        .WRB    03      ;OUTPUT THE BUFFER
        JSR @   SYSE1   ;SYSTEM ERROR
        JMP     0,3     ;NORMAL RETURN

MTAB1:  MTABF           ;REDUCED DATA OUTPUT BUFFER

MTASC:  040001          ;SPACE REVERSE COMMAND
MTAEC:  060000          ;EOF WRITE COMMAND
MTEOF:  000400          ;EOF ERROR MASK
MTACM:  050000+BDAC9    ;FF WRITE COMMAND
TMPCT:  BDACL           ;SIZE OF REDUCED DATA BUFFER ENTRY
TMPBK:  0               ;CURRENT TEMP.TM BLOCK #

NAMEA=  .*2
        .TXT    /MTO:6/
NAMEB=  .*2
        .TXT    /TEMPC.TM/

NAMEC=  .*2
        .TXT    /BDACS.DA/

DPMSG=  .*2
        .BLK    10      ;20 BYTES
                006400  ;<CR>
                0
```

AD-A049 303 HARRY DIAMOND LABS ADELPHI MD F/G 9/2
BDACS SOFTWARE.(U)
NOV 77 J C INGRAM MIPR-76628
UNCLASSIFIED HDL-TR-1831 NL

2 of 2
ADA049 303

END
DATE FILMED
3-78
DDC

```
MTABF=  .
        .BLK    BDAC9   ;REDUCED DATA OUTPUT BUFFER
        0

BUFFR=.
        .BLK    BDACM   ;BUFFER AREA
        0

        .END
```

APPENDIX A

A-6. Overlay Module No. 5--PHAS5.SR

NAME BLOCK NAME= PHAS5.SR

TIME BLOCK

```
        .TITL   PHAS5   ;JCI     19 MAR 76

        .TXTM   1

        .ENT    OVST5

        .EXTD   OVRTN ERRTN ERCOD MAGFG PRINT
        .EXTD   HSMON LSMON HLDCT SMPRT

        .EXTN   MSKTB LSBTB MEBPT

        .NREL

OVST5:  JSR @   OPFL1   ;OPEN
                NAMEG   ;'$TTI'
                02      ;ON CH #2
JP0:    JSR @   CREA1   ;CREATE A FILE
                NAMED   ;'TEMPD.TM'
        JSR @   OPFL1   ;OPEN
                NAMED   ;'TEMPD.TM'
                03      ;ON CH #3
        JSR @   OPFL1   ;OPEN
                NAMEJ   ;'POSTSCRIPT.DA'
                04      ;ON CH #4

LP1:    JSR @   RDLU2   ;READ A QUERY LINE
                JP1     ;EOF
        STA     2,TEMP1 ;HOLD THE BYTE POINTER
        DSZ     TEMP1   ;DEC THE POINTER
        LDA     2,UTBP1 ;UTILITY BYTE POINTER
        JSR @   GTBY1   ;GET A BYTE
        LDA     1,ASCAK ;<*>
        SUB     0,1     ;FLAG 0 IF <*>
        STA     1,TEMP2 ;HOLD THE FLAG
        LDA     2,TEMP1 ;RETRIEVE THE POINTER
        MOV #   1,1,SNR ;SKIP IF NOT <*>
        JMP     JP2
        SUB     0,0     ;CLEAR AC0
        JSR @   STBY1   ;REPLACE <CR> WITH <NUL>
JP2:    JSR @   TYLU1   ;ECHO THE LINE
        LDA     1,TEMP2 ;RETRIEVE FLAG
        MOV #   1,1,SNR ;SKIP IF RESPONSE REQUIRED
        JMP     JP3
        LDA     2,TEMP1 ;RETRIEVE POINTER
LP2:    JSR @   RDLU3   ;READ INTO UTILITY
                02      ;FROM CH #2
        ADCZL   0,0     ;FORCE -2
        ADD     0,2     ;BACK UP 2 BYTES
        STA     0,TEMP2 ;HOLD FLAG
```

```
        JSR @   GTBY1   ;GET A BYTE
        LDA     1,ASCCN ;<¬>
        SUB #   0,1,SZR ;SKIP IF <¬>
        JMP     JP3
        LDA     0,ASCCR ;<CR>
        JSR @   STBY1   ;REPLACE <¬> WITH <CR>
        ISZ     TEMP2   ;INC FLAG
JP3:    JSR @   WTLU1   ;WRITE THE LINE
        LDA     2,UTBP1 ;UTILITY BYTE POINTER
        ISZ     TEMP2   ;INC FLAG, SKIP IF <¬>
        JMP     LP1     ;READ NEXT QUERY
        JMP     LP2     ;CONTINUE QUERY RESPONSE

UTBP1:  UTBPT           ;POINTER TO UTILITY BUFFER
CREA1:  CREAT           ;CREATE A SEQ. FILE
OPMT1:  OPMTA           ;OPEN MAG TAPE FOR FF
OPFL1:  OPFLE           ;OPEN A CHANNEL
CLFL1:  CLFLE           ;CLOSE A CHANNEL
XFER1:  XFERF           ;TRANSFER ASCII FILES
PRLN1:  PRLNE           ;PRINT A LINE
RDLU1:  RDLUT           ;READ A LINE TO UTILITY
RDLU2:  RDLUR           ;READ A LINE, RETURN ON EOF
RDLU3:  RDLUA           ;READ A LINE, POINTER IN AC2
WTLU1:  WTLUT           ;WRITE A LINE FROM UTILITY
TYLU1:  TYLUT           ;TYPE A LINE FROM UTILITY
TYPM1:  TYPMG           ;TYPE A MESSAGE
GTSP1:  GTSPN           ;GET A SP #
GTBY1:  GTBYT           ;GET A BYTE
STBY1:  STBYT           ;STORE A BYTE
ERR03:  ERROR           ;ERROR ROUTINE
PRFL1:  PRFLE           ;PRINT THE FILE

TEMP1:  0               ;TEMPORARY
TEMP2:  0
ASCCR:  015             ;<CR>
ASCBK:  040             ;< >
ASCAK:  052             ;<*>
ASC00:  060             ;<0>
ASCIN:  116             ;<N>
ASCIY:  131             ;<Y>
ASCCN:  136             ;<¬>

DEV1:   NAMEH           ;'$TTO'
DEV2:   NAMEI           ;'$LPT'
OSETB:  BDACF           ;MUX INPUT MIN POINT #
OSETC:  BDACQ           ;LSB MIN POINT #

JP1:    JSR @   CLFL1   ;CLOSE
        04      ;CH #4
        JSR @   CLFL1   ;CLOSE
        03      ;CH #3
JP7:    JSR @   TYPM1   ;TYPE A MESSAGE
        MSG01   ;MESSAGE #1
        LDA     2,UTBP1 ;UTILITY
        JSR @   RDLU3   ;READ INTO UTILITY
        02      ;FROM CH #2
        LDA     2,UTBP1 ;UTILITY
```

APPENDIX A

```
        JSR @   GTBY1   ;GET A BYTE
        LDA     1,ASCIN ;<N>
        SUB #   0,1,SNR ;SKIP IF NOT <N>
        JMP     JP0
        LDA     1,ASCIY ;<Y>
        SUB #   0,1,SZR ;SKIP IF <Y>
        JMP     JP7
        JSR @   CLFL1   ;CLOSE
                02      ;CH #2
        LDA     0,MAGFG ;MAG TAPE FLAG
        MOV #   0,0,SNR ;SKIP IF MAG TAPE
        JMP     JP4
        JSR @   XFER1   ;TRANSFER
                NAMED   ;'TEMPD.TM'
                NAMEF   ;'M10:7'
JP4:    LDA     0,PRINT ;PRINT FLAG
        MOV #   0,0,SNR ;SKIP IF PRINT REQUIRED
        JMP @   OVRTN   ;RETURN TO ROOT BINARY
        LDA     1,DEV1  ;$TTO NAME POINTER
        MOVR #  0,0,SZC ;'SKIP IF TTY
        LDA     1,DEV2  ;$LPT NAME POINTER
        STA     1,.+2
        JSR @   OPFL1   ;OPEN
                0       ;DEVICE NAME
                02      ;ON CH #2
        JSR @   PRLN1   ;PRINT A LINE
                HEAD1   ;HEADER #1
                -2      ;<FF> AND 2 LINES
        JSR @   PRLN1   ;PRINT A LINE
                HEAD2   ;HEADER #2
                02      ;2 LINES
        JSR @   PRFL1   ;PRINT FILE
                NAMEA   ;PREAMBLE FILE
        JSR @   PRLN1   ;PRINT A LINE
                BKLN2   ;2 BLANK LINES
                02      ;2 LINES
        JSR @   PRFL1   ;PRINT FILE
                NAMED   ;POSTSCRIPT FILE
        JSR @   PRLN1   ;PRINT A LINE
                HEAD8   ;HEADER #3
                -2      ;<FF> AND 2 LINES
        JSR @   PRFL1   ;PRINT THE FILE
                METPT   ;METHOD FILE
        LDA     0,HSMON ;HSB MONITOR FLAG
        MOVL    0,0,SNC ;SKIP IF HSB PRINTOUT
        JMP     .+4
        MOVZR   0,0     ;RESET PRINT BIT
        STA     0,HSMON ;HOLD SAMPLE RATE
        JSR @   PRHS1   ;PRINT THE HSB
        SUB     0,0     ;CLEAR AC0
        JSR @   CHKP1   ;CHECK MUX BUFFER
PRTB1:          MSKTB   ;MUX TABLE
                BDACA/20;# MUX WORDS
        JSR @   CHKP1   ;CHECK LSB
PRTB2:          LSBTB   ;LSB TABLE
                BDACP/20;# LSB WORDS
        MOV #   0,0,SNR ;SKIP IF PRINTOUT REQUIRED
```

```
        JMP     JP5
        JSR @   OPFL1   ;OPEN
                NAMEL   ;'ASSIGNB.DA'
                04      ;ON CH #4
        JSR @   OPFL1   ;OPEN
                NAMEM   ;'ASSIGNC.DA'
                05      ;ON CH #5
        LDA     0,MAGFG ;MAG TAPE FLAG
        MOV #   0,0,SZR ;SKIP IF NO MAG TAPE
        JMP     .+5
        JSR @   OPFL1   ;OPEN
                NAMEC   ;'TEMPC.TM'
                03      ;ON CH #3
        JMP     LP0
        JSR @   OPMT1   ;OPEN MTA FOR FF
                NAMEE   ;'MT0:6'
                03      ;ON CH #3
LP0:    JSR @   PRLN2   ;PRINT A LINE
                HEAD3   ;HEADER #3
                -2      ;<FF> 2 LINES
        JSR @   PRLN2   ;PRINT A LINE
                HEAD4   ;HEADER #4
                02      ;ON CH #2
LP3:    JSR @   GTDA1   ;GET DATA POINTER
        STA     2,TEMP3 ;HOLD POINTER
        LDA @   0,TEMP3 ;STATE & SIGNAL WORD
        MOV #   0,0,SNR ;SKIP IF NOT END
        JMP     JP6
        LDA     1,LSBMK ;LSB MASK
        LDA     2,PRTB1 ;MUX PRINTOUT TABLE
        LDA     3,OSETB ;MUX INPUT MIN POINT #
        AND #   0,1,SNR ;SKIP IF LSB DATA WORD
        JMP     .+3
        LDA     2,PRTB2 ;LSB PRINTOUT TABLE
        LDA     3,OSETC ;LSB MIN POINT #
        LDA     1,SIGMK ;SIGNAL # MASK
        AND     1,0     ;GET SIGNAL #
        SUBZ    3,0,SNC ;SKIP IF AC3=<AC0
        JSR @   ERR03   ;ERROR ROUTINE
ERR4:   JSR @   FMMS1   ;FORM MASK AND DISPLACEMENT
        ADD     0,2     ;ADD DISPLACEMENT TO POINTER
        LDA     0,0,2   ;GET TABLE WORD
        AND #   1,0,SNR ;SKIP IF PRINTOUT REQUIRED
        JMP     LP3     ;LOOP BACK
        LDA     2,TEMP3 ;DATA POINTER
        LDA     0,1,2   ;MS TIME
        LDA     1,2,2   ;LS TIME
        LDA     2,LINE  ;LINE POINTER
        JSR @   FMDP1   ;FORM THE DP #
        LDA     0,ASCBK ;< >
        LDA @   3,TEMP3 ;STATE & SIGNAL WORD
        LDA     1,SEQMK ;SEQUENCE ERROR MASK
        AND #   3,1,SZR ;SKIP IF NO ERROR
        LDA     0,ASCIK ;<*>
        JSR @   STBY2   ;STORE THE ERROR FLAG BYTE
        LDA     0,ASCC0 ;<C>
        LDA     2,STPNT ;STATE BYTE POINTER
```

```
        LDA @   1,TEMP3 ;STATE & SIGNAL #
        MOVL #  1,1,SZC ;SKIP IF STATE 0
        INC     0,0     ;FORCE <1>
        JSR @   STBY2   ;STORE THE STATE
        LDA @   0,TEMP3 ;STATE & SIGNAL WORD
        LDA     1,SIGMK ;SIGNAL MASK
        LDA     2,SIGPT ;SIGNAL BYTE POINTER
        AND     1,0     ;MASK THE SIGNAL
        STA     0,ST1   ;HOLD THE SIGNAL
        JSR @   FMSP1   ;FORM THE SP #
        LDA @   0,TEMP3 ;STATE & SIGNAL WORD
        LDA     1,LSBMK ;ASSIGN MASK
        AND #   0,1,SZR ;SKIP IF ASSIGNB.DA
        JMP     .+4
        LDA     0,ASCIM ;<M>
        LDA     1,BPNTR ;B BUFFER POINTER
        JMP     .+3
        LDA     0,ASCIL ;<L>
        LDA     1,CPNTR ;C BUFFER POINTER
        STA     1,ST1+1 ;STORE BUFFER POINTER
        JSR @   STBY2   ;STORE ASSIGNMENT BYTE
        JSR @   OPEN1   ;GET ASSIGN FILE ASSIGNMENT POINTER
ST1:            0       ;POINT #
                0       ;BUFFER POINTER
        LDA     0,MNMPT ;MNEMONIC POINTER
        JSR @   MVBY1   ;MOVE THE BYTES
                20      ;20 BYTES MAX
        LDA     0,ASCIR ;<CR>
        JSR @   STBY2   ;STORE THE CR
        JSR @   PRLN2   ;PRINT THE LINE
LINE:           LNEBF   ;LINE BUFFER
                01      ;1 LINE
        JMP     LP3     ;LOOP BACK

JP6:    JSR @   CLFL2   ;CLOSE
                05      ;CH #5
        JSR @   CLFL2   ;CLOSE
                04      ;CH #4
        JSR @   CLFL2   ;CLOSE
                03      ;CH 33
JP5:    JSR @   CLFL2   ;CLOSE
                02      ;CH #2
        JSR @   CLFL2   ;CLOSE
                01      ;CH #1
        JMP @   OVRTN   ;RETURN TO ROOT BINARY

TEMP3:  0               ;TEMPORARY

PRLN2:  PRLNE           ;PRINT A LINE
GTDA1:  GTDAT           ;GET DATA WORD POINTER
FMSP1:  FMSPN           ;FORM SP #
FMDP1:  FMDPN           ;FORM DP #
FMMS1:  FMMSK           ;FORM MASK AND DISPLACEMENT
MVBY1:  MVBYT           ;MOVE A BYTE STRING
CHKP1:  CHKPM           ;CHECK PRINT MASKS
OPEN1:  OPEN            ;READ ASSIGNMENT FILE
PRHS1:  PRHSB           ;PRINT HSB
```

```
STBY2:  STBYT           ;STORE A BYTE
MVBY2:  MVBYT           ;MOVE BYTE STRING
OPFL2:  OPFLE           ;OPEN A CHANNEL
CLFL2:  CLFLE           ;CLOSE A CHANNEL
BPNTR:  BUFRB           ;B BUFFER POINTER
CPNTR:  BUFRC           ;C BUFFER POINTER

ASCIR:  015             ;<CR>
ASCIK:  052             ;<*>
ASCCO:  060             ;<0>
ASCIL:  114             ;<L>
ASCIM:  115             ;<M>
SIGMK:  000377          ;SIGNAL MASK
LSBMK:  040000          ;LSB MASK
SEQMK:  000400          ;SEQUENCE ERROR MASK
C20:    20

STPNT:  LNEBF+17        ;STATE POINTER
SIGPT:  LNEBF+24        ;SIGNAL POINTER
MNMPT:  LNEBF+35        ;MNMEMONIC POINTER

PRHSB:  STA     3,HSRTN ;RETURN ADDRS
        JSR @   OPFL2   ;OPEN
                NAMEB   ;'TEMPB.TM'
                04      ;ON CH #4
        JSR @   OPFL2   ;OPEN
                NAMEK   ;'ASSIGNA.DA'
                05      ;ON CH #5
        JSR @   PRLN2   ;PRINT
                HEAD5   ;HEADER #5
                -2      ;<FF> AND 2 LINES
        JSR @   PRLN2   ;PRINT
                HEAD6   ;HEADER #6
                02      ;2 LINES
        LDA     0,C20   ;HSB WORD SIZE
        STA     0,HSCNT ;SET COUNT
HSBAA:  LDA     0,ASCNT ;CURRENT BIT #
        LDA     2,ASNPT ;ASSIGNMENT LINE POINTER
        JSR @   FMSP1   ;FORM SP #
        JSR @   OPEN1   ;GET ASSIGN FILE ASSIGNMENT POINTER
ASCNT:          BDACS   ;BIT #
                BUFRA   ;BUFFER POINTER
        LDA     0,ASMNP ;MNEMONIC POINTER
        JSR @   MVBY2   ;MOV BYTE STRING
                20      ;20 BYTES MAX
        LDA     0,ASCIR ;<CR>
        JSR @   STBY2   ;STORE <CR>
        JSR @   PRLN2   ;PRINT A LINE
ASNPT:          ASGBF   ;ASSIGNMENT LINE POINTER
                01      ;1 LINE
        ISZ     ASCNT   ;INC BIT #
        DSZ     HSCNT   ;SKIP WHEN DONE
        JMP     HSBAA
        JSR @   PRLN2   ;PRINT A LINE
                HEAD7   ;HEADER #7
                -2      ;<FF> AND 2 LINES
HSBJO:  LDA     2,HSBPT ;HSB POINTER
```

APPENDIX A

```
        DSZ     HSBCT   ;HSB COUNT
        JMP     HSBJ1
        JSR @   RDBL1   ;READ A BLOCK
BUFPT:          BUFRB   ;BUFFER POINTER
                0       ;CURRENT BLOCK #
                404     ;SIZE & CH #
                HSBJ3   ;EOF ADDRS
        LDA     2,HSBSZ ;HSB BUFFER SIZE
        STA     2,HSBCT ;RESET COUNT
        LDA     2,BUFPT ;BUFFER POINTER
        STA     2,HSBPT ;HSB POINTER
HSBJ1:  ISZ     HSBPT   ;INC THE POINTER
        LDA     0,0,2   ;GET BUFFER WORD
        STA     0,TEMP3 ;HOLD THE WORD
        LDA     0,HSTIM ;HS TIME
        LDA     1,HSTIM+1
        MOV     0,2     ;MOV MS TIME
        LDA     3,HSMON ;HSB SAMPLE RATE
        ADDZ    1,3,SZC ;ADD IN SAMPLE RATE
        INC     2,2
        STA     2,HSTIM ;RESTORE UPDATED TIME
        STA     3,HSTIM+1
        LDA     2,HSB   ;LINE POINTER
        JSR @   FMDP1   ;FORM DP #
        LDA     0,C20   ;HSB WORD SIZE
        STA     0,HSCNT ;STORE COUNT
HSBJ2:  LDA     0,ASCC0 ;<0>
        LDA     1,TEMP3 ;HSB WORD
        MOVZR   1,1,SZC ;SKIP IF 0 BIT
        INC     0,0     ;FORCE <1>
        STA     1,TEMP3 ;HOLD SHIFTED WORD
        INC     2,2     ;INC POINTER
        INC     2,2
        JSR @   STBY2   ;STORE THE BYTE
        DSZ     HSCNT   ;SKIP WHEN DONE
        JMP     HSBJ2
        JSR @   PRLN2   ;PRINT A LINE
HSB:            HSBBF   ;HSB LINE
                01      ;1 LINE
        JMP     HSBJ0   ;LOOP BACK
HSBJ3:  JSR @   CLFL2   ;CLOSE
                04      ;CH #4
        JSR @   CLFL2   ;CLOSE
                05      ;CH #5
        JMP @   HSRTN   ;RETURN

HSTIM:  0               ;CURRENT TIME
        0
HSCNT:  0               ;COUNTER
HSBCT:  01              ;BUFFER COUNT
HSBSZ:  400             ;BUFFER SIZE
HSBPT:  BUFRB+400       ;BUFFER POINTER
ASMNP:  ASCBF+7         ;ASSIGNMENT MNEMONIC POINTER
HSRTN:  0               ;RETURN ADDRS

TMPI1:  TMPIN           ;REDUCED DATA FROM DISK
MTAI1:  MTAIN           ;REDUCED DATA FROM MAG TAPE
```

```
RDBL1:  RDBLK           ;READ A BLOCK

GTDRT:  0               ;RETURN ADDRS
GTDAT:  STA     3,GTDRT ;STORE RETURN
        DSZ     MTACT   ;SKIP IF BUFFER COUNT 0
        JMP     GTDJ1
        LDA     1,MAGFG ;MAG TAPE FLAG
        MOV #   1,1,SZR ;SKIP IF NO MAG TAPE
        JMP     .+4
        JSR @   TMPI1   ;DISK INPUT
                JP6     ;EOF CONDITION
        JMP     .+3
        JSR @   MTAI1   ;MAG TAPE INPUT
                JP6     ;EOF CONDITION
        LDA     1,MTAST ;START OF INPUT BUFFER
        STA     1,MTAPT ;RESET POINTER
        LDA     1,MTASZ ;BUFFER COUNT
        STA     1,MTACT ;RESET COUNT
GTDJ1:  LDA     2,MTAPT ;GET CURRENT POINTER
        LDA     0,MTAOF ;OFFSET
        ADD     2,0     ;ADD OFFSET
        STA     0,MTAPT ;UPDATE POINTER
        JMP @   GTDRT   ;RETURN

MTAST:  BUFFR           ;INPUT BUFFER POINTER
MTAPT:  BUFFR+BDAC9     ;CURRENT POINTER
MTASZ:  400             ;BUFFER SIZE
MTACT:  01              ;CURRENT COUNT
MTAOF:  BDACL           ;DATA ENTRY SIZE


PH5CD:  2400            ;PHASE 5 ERROR CODE
ERTBL:  .+1             ;ERROR TABLE POINTER
        ERR1            ;ASSIGN FILE READ ERROR
        ERR2            ;ASSIGN FILE SEQUENCE ERROR
        ERR3            ;LINE COUNT ERROR
        ERR4            ;MUX OR LSB POINT # OFB
        -1              ;END OF TABLE
ERROR:  LDA     0,PH5CD ;PHASE 5 ERROR CODE
        LDA @   2,ERTBL ;TABLE ENTRY
        INC     0,0     ;INC ERROR COUNT
        ISZ     ERTBL   ;INC POINTER
        COM #   2,2,SNR ;SKIP IF NOT EOT
        JMP     .+3
        SUB #   2,3,SZR ;SKIP IF ERROR FOUND
        JMP     .-6     ;LOOP BACK
        JMP     .+2
SYSER:  ADC     0,0     ;FORCE -1
        STA     0,ERCOD ;STORE IN ROOT BINARY
        JMP @   ERRTN   ;ERROR RETURN TO ROOT BINARY

ERR01:  ERROR           ;ERROR ROUTINE
OPSIZ:  40              ;# MNEMONICS PER BLOCK
OPOFF:  0               ;POINTER OFFSET
OPRTN:  0               ;RETURN ADDRS
OPEN:   STA     3,OPRTN ;RETURN ADDRS
        LDA     1,0,3   ;POINT #
```

```
         LDA     3,1,3   ;BUFFER POINTER
         LDA     0,-3,3  ;INITIAL POINT #
         SUBZ    0,1,SNC ;SKIP IF AC0=<AC1
         JSR @   ERR01   ;ERROR
ERR2:    SUB     0,0     ;CLEAR AC0
         LDA     2,OPSIZ ;# OF MNEMONICS PER BLOCK
         DIV
         ADDZL   0,0     ;*4
         MOVZL   0,0     ;*2
         STA     0,OPOFF ;STORE POINTER OFFSET
         LDA     2,-1,3  ;CURRENT BLOCK #
         STA     1,-1,3  ;STORE NEW BLOCK #
         MOV     3,0     ;MOV BUFFER POINTER
         SUB #   2,1,SNR ;SKIP IF NOT SAME
         JMP     .+5
         LDA     2,-2,3  ;GET CH # AND COUNT
         .SYSTM
         .RDB    77      ;READ A BLOCK
         JSR @   ERR01   ;ERROR
ERR1:    LDA     2,OPOFF ;GET OFFSET
         ADDZL   0,2     ;ADD IN BASE, FORM BYTE POINTER
         LDA     3,OPRTN ;RETURN ADDRS
         JMP     2,3     ;RETURN

C11:     11
C12:     12
CRRTN:   0               ;RETURN ADDRESS
CREAT:   LDA     0,0,3   ;NAME POINTER
         INC     3,3
         STA     3,CRRTN ;STORE RETURN
CREAA:   .SYSTM
         .CREA           ;CREATE SEQ. FILE
         JMP     .+2     ;ERROR
         JMP @   CRRTN   ;RETURN
         LDA     1,C11
         SUB #   1,2,SZR ;SKIP IF FILE EXISTS
         JMP @   SYSE1   ;ERROR
         JSR     DELEA   ;DELETE THE FILE
         JMP     CREAA   ;TRY AGAIN

DELET:   LDA     0,0,3   ;NAME POINTER
         INC     3,3
DELEA:   STA     3,USP   ;STORE RETURN
         .SYSTM
         .DELET          ;DELETE THE FILE
         JMP     .+2
         JMP     0,3     ;NORMAL RETURN
         LDA     1,C12
         SUB #   1,2,SZR ;SKIP IF NO FILE
         JMP @   SYSE1   ;ERROR
         JMP     0,3     ;NORMAL RETURN

XFRTN:   0               ;RETURN ADDRESS
XFERF:   LDA     0,0,3   ;SOURCE FILE
         LDA     1,1,3   ;DESTINATION FILE
         STA     0,XFSFL
         STA     1,XFDFL
```

```
        STA     3,XFRTN ;STORE RETURN
        JSR     OPFLE   ;OPEN SOURCE FILE
XFSFL:          0       ;POINTER
                04      ;ON CH#4
        JSR     OPFLE   ;OPEN DESTINATION FILE
XFDFL:          0       ;POINTER
                03      ;ON CH#3
        JSR     RDLUR   ;READ A LINE
                .+3     ;RETURN ON EOF
        JSR     WTLUT   ;WRITE A LINE
        JMP     .-3     ;LOOP BACK
        JSR     CLFLE   ;CLOSE
                04      ;CH#4
        JSR     CLFLE   ;CLOSE
                03      ;CH#3
        LDA     3,XFRTN ;RETURN ADDRESS
        JMP     2,3     ;RETURN

OPFLE:  LDA     0,0,3   ;NAME POINTER
        LDA     2,1,3   ;CH #
        SUB     1,1     ;USE DEFAULT CHARACTERISTICS
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .OPEN   77      ;OPEN THE CHANNEL
        JMP @   SYSE1   ;ERROR
        JMP     2,3     ;NORMAL RETURN

CLFLE:  LDA     2,0,3   ;CH #
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .CLOS   77      ;CLOSE THE CHANNEL
        JMP @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

TYPMG:  LDA     0,0,3   ;MSG POINTER
        INC     3,3
        SUBZL   2,2     ;FORCE A +1 FOR CH#
        JMP     .+5
TYLUT:  SUBZL   2,2     ;FORCE A +1 FOR CH#
        JMP     .+2
WTLUT:  LDA     2,C03   ;CH #3, BY DEFAULT
        LDA     0,UTBPO ;UTILITY BYTE POINTER
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .WRL    77      ;WRITE A LINE
        JMP @   SYSE1   ;ERROR
        JMP     0.3     ;NORMAL RETURN
RDSUT:  LDA     1,0,3   ;BYTE COUNT
        LDA     0,UTBPO ;UTILITY BYTE POINTER
        LDA     2,C04   ;CH #4
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .RDS    77      ;READ SEQUENTIAL BYTES
        JSR @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN
```

APPENDIX A

```
SYSE1:  SYSER           ;SYSTEM ERROR

C03:    03
C04:    04
C06:    06
RDRTN:  0
RDLUA:  ADC     1,1     ;RETURN FLAG -1
        MOV     2,0     ;BYTE POINTER
        LDA     2,0,3   ;CH #
        INC     3,3
        JMP     .+7
RDLUR:  LDA     1,0,3   ;RETURN ADDRESS
        INC     3,3
        JMP     .+2
RDLUT:  ADC     1,1     ;RETURN FLAG -1
        LDA     0,UTBPO ;UTILITY BYTE POINTER
        LDA     2,C04   ;CH #4
        STA     1,RDRTN ;STORE RETURN
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .RDL    77      ;READ A LINE
        JMP     .+4     ;CHECK FOR EOF
        MOV     0,2     ;MOVE THE BYTE POINTER
        ADD     1,2     ;OFFSET THE POINTER
        JMP     0,3     ;NORMAL RETURN
        LDA     0,C06   ;EOF CODE
        SUB #   0,2,SZR ;SKIP IF EOF
        JMP @   SYSE1   ;ERROR
        LDA     0,RDRTN ;RETURN
        COM #   0,0,SZR ;SKIP IF NO EOF RETURN
        JMP @   RDRTN   ;EOF RETURN
        JMP @   SYSE1   ;ERROR

SPLDS:  LDA     0,0,3   ;DEVICE BYTE POINTER
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .SPDA           ;DISABLE SPOOLING
        JSR @   SYSE1   ;ERROR
        JMP     1,3     ;NORMAL RETURN

WTLNE:  LDA     0,0,3   ;NAME POINTER
        LDA     2,1,3   ;CH #
        STA     3,USP   ;STORE RETURN
        .SYSTM
        .WRL    77      ;WRITE A LINE
        JSR @   SYSE1   ;SYSTEM ERROR
        JMP     2,3     ;NORMAL RETURN

PRFRT:  0               ;RETURN ADDRS
PRFLE:  LDA     0,0,3   ;NAME POINTER
        INC     3,3
        STA     3,PRFRT ;STORE RETURN
        STA     0,.+2   ;STORE POINTER
        JSR     OPFLE   ;OPEN
                0       ;NAME POINTER
                04      ;ON CH #4
```

```
        JSR     RDLUR   ;READ A LINE
                .+5     ;EOF RETURN
        JSR     PRLNE   ;PRINT THE LINE
                UTBPT   ;UTILITY
                01      ;1 LINE
        JMP     .-5
        JSR     CLFLE   ;CLOSE
                04      ;CH #4
        JMP @   PRFRT   ;RETURN

UTBPO:  UTBPT           ;UTILITY BYTE POINTER

SKFLG:  -1              ;LINE SKIP FLAG
LNMAX:  036             ;MAX LINE COUNT
LNCNT:  20              ;CURRENT LINE COUNT
PGCNT:  0               ;CURRENT PAGE COUNT
PRSTR:  0               ;LINES STORAGE
PRRTN:  0               ;RETURN ADDRS
ERRO2:  ERROR           ;ERROR ROUTINE
FMSP2:  FMSPN           ;FORM SP #
PGOFF:  PAGE1+13
PRLNE:  LDA     0,0,3   ;MESSAGE POINTER
        LDA     1,1,3   ;# LINES
        STA     3,PRRTN ;RETURN ADDRS
        STA     1,PRSTR ;STORE LINES
        STA     0,PRMSG ;STORE MESSAGE POINTER
        MOVL #  1,1,SNC ;SKIP IF NEG
        JMP     .+6
        ADC     0,0     ;FORCE -1
        STA     0,SKFLG ;RESET SKIP FLAG
        NEG     1,1     ;FORM POSITIVE
        STA     1,PRSTR ;STORE POS LINE COUNT
        JMP     PRLNC
        LDA     0,SKFLG ;SKIP LINE FLAF
        READS   2       ;READ SWITCHES
        COM #   0,0,SNR ;SKIP IF FLAG SET
        JMP     .+3
        COM #   2,2,SNR ;SKIP IF NOT ALL UP
        JMP     2,3     ;RETURN
        COM #   2,2,SZR ;SKIP IF ALL UP
        STA     2,SKFLG ;SET SKIP FLAG
        LDA     0,LNCNT ;CURRENT LINE COUNT
        SUBZ #  1,0,SZC ;SKIP IF AC1>AC0
        JMP     PRLNA
PRLNC:  LDA     0,PRINT ;PRINT FLAG
        MOVR #  0,0,SZC ;SKIP IF $TTO
        JMP     PRLNB
        ISZ     LNCNT   ;INC LINE COUNT
        ISZ     LNCNT   ;AGAIN
        JSR     WTLNE   ;WRITE A LINE
                BKLN1   ;1 BLANK LINE
                02      ;ON CH #2
        DSZ     LNCNT   ;SKIP WHEN DONE
        JMP     .-4
        JMP     .+4
PRLNB:  JSR     WTLNE   ;WRITE A LINE
                TOF     ;TOP OF FORM
```

APPENDIX A

```
                02          ;ON CH #2
        LDA     0,LNMAX     ;MAX COUNT
        STA     0,LNCNT     ;RESET COUNT
        ISZ     PGCNT       ;INC PAGE COUNT
        LDA     0,PGCNT     ;GET PAGE #
        LDA     2,PGOFF     ;PAGE # OFFSET
        JSR @   FMSP2       ;FORM SP #
        JSR     WTLNE       ;WRITE A LINE
                PAGE1       ;PAGE #
                02          ;ON CH #2
PRLNA:  JSR     WTLNE       ;WRITE A LINE
PRMSG:          0           ;MESSAGE POINTER
                02          ;ON CH #2
        LDA     1,PRSTR     ;LINE COUNT STORAGE
        LDA     0,LNCNT     ;CURRENT COUNT
        SUBZ    1,0,SNC     ;SKIP IF AC1=<AC0
        JSR @   ERR02       ;ERROR ROUTINE
ERR3:   STA     0,LNCNT     ;UPDATE COUNT
        LDA     3,PRRTN     ;RETURN ADDRS
        JMP     2,3         ;RETURN

MVRTN:  0
MVWUT:  LDA     2,0,3       ;DESTINATION POINTER
        LDA     1,1,3       ;COUNT
        STA     3,MVRTN     ;STORE RETURN
        LDA     3,UTBPO     ;UTILITY BYTE POINTER
        MOVZR   3,3         ;FORM ADDRESS
        NEG     1,1         ;NEG THE COUNT
        LDA     0,0,3       ;GET WORD
        STA     0,0,2       ;STORE WORD
        INC     3,3
        INC     2,2
        INC     1,1,SZR     ;SKIP WHEN DONE
        JMP     .-5
        LDA     3,MVRTN     ;RETURN ADDRESS
        JMP     2,3         ;RETURN

CHKCT:  0                   ;COUNTER
CHKPM:  LDA     2,0,3       ;TABLE POINTER
        LDA     1,1,3       ;COUNT
        ADD     1,2         ;OFFSET THE POINTER
        ADD     1,2         '
        STA     2,0,3       ;UPDATE THE TABLE POINTER
        STA     1,CHKCT     ;STORE THE COUNT
        LDA     1,0,2       ;GET A WORD
        INC     2,2         ;INC POINTER
        MOV #   1,1,SZR     ;SKIP IF 0
        INC     0,0         ;SET FLAG
        DSZ     CHKCT       ;SKIP WHEN DONE
        JMP     .-5
        JMP     2,3         ;RETURN

FMMSK:  SUBZL   1,1         ;SET BIT TO +1
        MOVZR   0,0,SZC     ;SKIP IF NOT 2**0
        MOVZL   1,1         ;SHIFT 1 PLACE LEFT
        MOVZR   0,0,SZC     ;SKIP IF NOT 2**1
        ADDZL   1,1         ;SHIFT 2 PLACE LEFT
```

```
        MOVZR   0,0,SNC ;SKIP IF     2**2
        JMP     .+3
        ADDZL   1,1     ;SHIFT 4 PLACE LEFT
        ADDZL   1,1
        MOVZR   0,0,SZC ;SKIP IF NOT 2**3
        MOVS    1,1     ;SHIFT 8 PLACE LEFT
        JMP     0,3     ;RETURN
GTDPN:  ADC     0,0     ;DP FLAG
        MOV     0,1     ;ERROR ON NO # FLAG
        JMP     .+6
GTSPR:  LDA     1,0,3   ;EXIT POINTER IF NO #
        INC     3,3
        JMP     .+2
GTSPN:  ADC     1,1     ;ERROR ON NO # FLAG
        SUBZL   0,0     ;SP FLAG
        STA     0,GTFGO ;SP/DP FLAG
        SUB     0,0     ;CLEAR AC0
        STA     0,GTFG1 ;# FOUND FLAG
        STA     0,GTOVF ;OVERFLOW FLAG
        STA     0,GTSTR ;# STORAGE
        STA     0,GTSTR+1
        STA     1,GTERT ;ERROR EXIT FLAG
        STA     2,GTBPT ;BYTE POINTER
        STA     3,GTRTN ;NORMAL RETURN
        JMP     .+2
GTLP1:  ISZ     GTBPT   ;INC BYTE POINTER
        LDA     2,GTBPT ;GET THE POINTER
        JSR     GTBYT   ;GET THE BYTE
        LDA     1,ASC1C ;<CR>
        SUB     0,1,SNR ;SKIP IF NOT <CR>
        JMP     GTLP2   ;DONE
        LDA     1,ASC19 ;<9>
        ADCZ #  1,0,SZC ;SKIP IF AC1>=AC0
        JMP     GTLP2   ;NOT A DIGIT
        LDA     1,ASC10 ;<0>
        ADCZ #  0,1,SZC ;SKIP IF AC0>=AC1
        JMP     GTLP2   ;NOT A DIGIT
        ISZ     GTFG1   ;INC # FOUND FLAG
        SUB     1,0     ;FORM BINARY
        STA     0,GTUTL ;HOLD THE #
        LDA     0,GTSTR ;GET PREVIOUS VALUE
        LDA     1,GTSTR+1
        MOVZL   1,1     ;N*2
        MOVL    0,0,SZC ;SKIP IF NO OVERFLOW
        ISZ     GTOVF   ;INC FLAG
        MOVZL   1,3     ;N*4
        MOVL    0,2,SZC ;SKIP IF NO OVERFLOW
        ISZ     GTOVF   ;INC FLAG
        MOVZL   3,3     ;N*8
        MOVL    2,2,SZC ;SKIP IF NO OVERFLOW
        ISZ     GTOVF   ;INC FLAG
        ADDZ    1,3,SZC ;N*10
        INC     0,0
        ADDZ    0,2,SZC ;SKIP IF NO OVERFLOW
        ISZ     GTOVF   ;INC FLAG
        LDA     1,GTUTL ;RETRIEVE THE BINARY
        ADDZ    1,3,SNC ;ADD TO PREVIOUS #
```

APPENDIX A

```
        JMP     .+3
        INCZ    2,2,SZC ;SKIP IF NO OVERFLOW
        ISZ     GTOVF   ;INC FLAG
        STA     2,GTSTR ;HOLD UPDATED #
        STA     3,GTSTR+1
        JMP     GTLP1   ;GET NEXT BYTE
GTLP2:  LDA     3,GTFG1 ;# FOUND FLAG
        MOV #   3,3,SNR ;SKIP IF # FOUND
        JMP     GTLP3
        LDA     0,GTSTR ;RETRIEVE THE #
        LDA     1,GTSTR+1
        LDA     2,GTBPT ;RETRIEVE THE BYTE POINTER
        LDA     3,GTOVF ;OVERFLOW FLAG
        DSZ     GTFGO   ;SKIP IF SP #
        JMP     .+4
        MOV #   0,0,SZR ;SKIP IF NOT SP OVERFLOW
        INC     3,3,SKP ;INC FLAG
        MOV     1,0     ;MOVE SP #
        MOV #   3,3,SZR ;SKIP IF NO OVERFLOW
        JSR @   GTERR   ;ERROR ROUTINE
ERR8:   JMP @   GTRTN   ;NORMAL RETURN
GTLP3:  MOV #   1,1,SZR ;SKIP IF EOL
        JMP     GTLP1   ;GET NEXT BYTE
        LDA     1,GTERT ;ERROR RETURN FLAG
        COM #   1,1,SNR ;SKIP IF ADDRESS SPECIFIED
        JSR @   GTERR   ;ERROR ROUTINE
ERR9:   JMP @   GTERT   ;EOL RETURN, NO # FOUND

GTERR:  ERROR           ;ERROR ROUTINE
GTRTN:  0               ;RETURN ADDRESS
GTERT:  0               ;EOL RETURN FLAG
GTBPT:  0               ;BYPE POINTER STORAGE
GTSTR:  0               ;# STORAGE
        0
GTFGO:  0               ;DP/SP FLAG
GTFG1:  0               ;# FOUND FLAG
GTOVF:  0               ;OVERFLOW FLAG
GTUTL:  0               ;UTILITY
ASCIC:  015             ;<CR>
ASCIB:  040             ;< >
ASCI9:  071             ;<9>
ASCI0:  060             ;<0>

GTBYT:  LDA     1,BTMSK ;BYTE MASK
        MOVZR   2,2,SNC ;FORM WORD ADDRESS, SKIP IF RHS
        MOVS    1,1     ;SWAP THE MASK
        LDA     0,0,2   ;GET WORD
        AND     1,0,SNC ;MASK THE WORD, SKIP IF RHS
        MOVS    0,0     ;SWAP THE WORD
        MOVL    2,2     ;RESTORE BYTE POINTER
        JMP     0,3     ;RETURN

BTMSK:  377             ;BYTE MASK

STRTN:  0
STBYT:  LDA     1,BTMSK ;BYTE MASK
        AND     1,0     ;MASK THE WORD
```

```
        MOVZR   2,2,SNC ;FORM WORD ADDRESS, SKIP IF RHS
        MOVS    0,0,SZC ;SWAP WORD, SKIP IF LHS
        MOVS    1,1     ;SWAP MASK
        STA     3,STRTN ;STORE RETURN
        LDA     3,0,2   ;GET WORD
        AND     1,3     ;MASK THE WORD
        ADD     0,3     ;ADD IN NEW BYTE
        STA     3,0,2   ;RESTORE THE WORD
        MOVL    2,2     ;RESTORE BYTE POINTER
        INC     2,2     ;INC POINTER
        JMP @   STRTN   ;RETURN

MVSPT:  0               ;SOURCE POINTER
MVDPT:  0               ;DESTINATION POINTER
MVCNT:  0               ;COUNTER
MVBRT:  0               ;RETURN ADDRS
MVBYT:  LDA     1,0,3   ;MAX BYTES
        INC     3,3
        STA     0,MVDPT ;DESTINATION POINTER
        STA     1,MVCNT ;MAX COUNT
        STA     2,MVSPT ;SOURCE POINTER
        STA     3,MVBRT ;RETURN ADDRS
MVBYA:  LDA     2,MVSPT ;GET SOURCE POINTER
        ISZ     MVSPT
        JSR     GTBYT   ;GET THE BYTE
        LDA     2,MVDPT ;GET DESTINATION POINTER
        ISZ     MVDPT
        MOV #   0,0,SNR ;SKIP IF NOT NUL
        JMP @   MVBRT   ;RETURN
        JSR     STBYT   ;STORE THE BYTE
        DSZ     MVCNT   ;SKIP IF MAX COUNT
        JMP     MVBYA   ;LOOP BACK
        JMP @   MVBRT   ;RETURN

FMSPN:  STA     3,FMRTN ;STORE RETURN
        LDA     3,FMSPP ;SP TABLE POINTER
        MOV     0,1     ;MOV LS WORD
        SUB     0,0     ;CLEAR MS WORD
        JMP     .+3
FMDPN:  STA     3,FMRTN ;STORE RETURN
        LDA     3,FMDPP ;DP TABLE POINTER
        STA     3,FMTBP ;HOLD THE POINTER
        STA     2,FMBPT ;HOLD BYTE POINTER
        STA     1,FMSTR+1
        STA     0,FMSTR ;HOLD DP #
        SUB     0,0     ;CLEAR ACO
        STA     0,FMSUP ;RESET ZERO SUPPRESSION FLAG
FMJPO:  LDA @   2,FMTBP ;POWER TABLE ENTRY
        COM #   2,2,SZR ;SKIP IF EOT
        JMP     .+3
        LDA     2,FMBPT ;RETRIEVE BYTE POINTER
        JMP @   FMRTN   ;RETURN
        ISZ     FMTBP   ;INC POINTER
        LDA @   3,FMTBP ;POWER TABLE ENTRY
        ISZ     FMTBP   ;INC POINTER
        MOVR #  3,3,SNR ;SKIP IF NOT LAST ENTRY
        ISZ     FMSUP   ;SET SUPPRESSION FLAG
```

APPENDIX A

```
          SUB     0,0       ;CLEAR AC0
          STA     0,FMCNT   ;RESET COUNT
          LDA     0,FMSTR   ;RETRIEVE DP #
          LDA     1,FMSTR+1
          SUBZ    3,1,SNC   ;PERFORM DP SUBTRACT
          ADC     2,0,SKP
          SUB     2,0
          MOVL #  0,0,SZC   ;SKIP IF NO OVERFLOW
          JMP     .+3
          ISZ     FMCNT     ;INC DIGIT COUNT
          JMP     .-6
          ADDZ    3,1,SZC   ;POSITIVE
          INC     0,0
          ADD     2,0
          STA     0,FMSTR   ;HOLD THE REMAINDER
          STA     1,FMSTR+1
          LDA     0,ASCIO   ;<0>
          LDA     1,FMCNT   ;DIGIT COUNT
          MOV #   1,1,SZR   ;SKIP IF NO COUNT
          ISZ     FMSUP     ;SET SUPPRESSION FLAG
          ADD     1,0       ;ADD COUNT TO BASE
          LDA     1,FMSUP   ;SUPPRESSION FLAG
          MOV #   1,1,SNR   ;SKIP IF NO SUPPRESSION
          LDA     0,ASCIB   ;< >
          LDA     2,FMBPT   ;BYTE POINTER
          ISZ     FMBPT     ;INC THE POINTER
          JSR     STBYT     ;STORE THE BYTE
          JMP     FMJPO     ;LOOP BACK

FMRTN:    0                 ;RETURN ADDRS
FMTBP:    0                 ;POWER OF TEN TABLE POINTER
FMBPT:    0                 ;BYTE POINTER
FMSUP:    0                 ;SUPPRESSION FLAG
FMCNT:    0                 ;DIGIT COUNT
FMSTR:    0                 ;DP # STORAGE
          0
FMSPP:    .+14              ;START OF SP TABLE
FMDPP:    .+1               ;START OF DP TABLE
          035632            ;10**9
          145000
          002765            ;10**8
          160400
          000230            ;10**7
          113200
          000017            ;10**6
          041100
          000001            ;10**5
          103240
          000000            ;10**4
          023420
          000000            ;10**3
          001750
          000000            ;10**2
          000144
          000000            ;10**1
          000012
          000000            ;10**0
          000001
          177777            ;EOT
```

```
RDBLK:  STA     3,USP    ;STORE RETURN
        LDA     0,BLKMK  ;BLOCK MASK
        LDA     1,1,3    ;CURRENT BLOCK
        LDA     2,2,3    ;SIZE & CH #
        ANDS    2,0      ;GET SIZE
        ADD     1,0      ;ADD TO CURRENT BLOCK
        STA     0,1,3    ;UPDATE CURRENT BLOCK
        LDA     0,0,3    ;BUFFER POINTER
        LDA     3,3,3    ;EOF ADDRS
        STA     3,EFRTN  ;EOF RETURN
        .SYSTM
        .RDB    77       ;READ THE BLOCK
        JMP     .+2
        JMP     4,3      ;RETURN
        LDA     1,EOFCD  ;EOF CODE
        SUB #   1,2,SZR  ;SKIP IF EOF
        JSR @   SYSE2    ;SYSTEM ERROR
        JMP @   EFRTN    ;EOF RETURN

TMPRT:  0                ;RETURN ADDRS
TMPIN:  LDA     0,0,3    ;EOF RETURN
        INC     3,3
        STA     3,TMPRT  ;RETURN ADDRS
        STA     0,.+5
        JSR     RDBLK    ;READ A BLOCK
MTAB1:          BUFFR    ;BUFFER POINTER
                0        ;BLOCK #
                BDACL*400+03
                0        ;EOF RETURN
        JMP @   TMPRT

BLKMK:  177400           ;BLOCK SIZE MASK
EOFCD:  06               ;EOF CODE
EFRTN:  0                ;EOF RETURN
EOFMK:  000400           ;EOF MASK
SYSE2:  SYSER            ;SYSTEM ERROR
RDCMD:  000000+8DAC9     ;FF READ COMMAND

MTAIN:  LDA     0,0,3    ;EOF RETURN
        STA     0,EFRTN  ;STORE THE RETURN
        LDA     0,MTAB1  ;INPUT DATA BUFFER POINTER
        LDA     1,RDCMD  ;READ COMMAND
        STA     3,USP    ;STORE RETURN
        .SYSTM
        .MTDIO  03       ;INPUT BUFFER FORM MAG TAPE
        JMP     .+2      ;ERROR
        JMP     1,3      ;NORMAL RETURN
        LDA     1,EOFMK  ;EOF MASK
        AND #   1,2,SNR  ;SKIP IF EOF
        JSR @   SYSE2    ;SYSTEM ERROR
        JMP @   EFRTN    ;EOF RETURN

OPMTA:  LDA     0,0,3    ;NAME POINTER
        LDA     2,1,3    ;CH #
        SUB     1,1      ;DEFAULT CHARACTERISTICS
        STA     3,USP    ;STORE RETURN
        .SYSTM
```

APPENDIX A

```
        .MTOPD  77      ;OPEN THE CHANNEL
        JSR @   SYSE2   ;SYSTEM ERROR
        JMP     2,3     ;NORMAL RETURN

NAMEA=  .*2
        .TXT    /TEMPA.TM/

NAMEB=  .*2
        .TXT    /TEMPB.TM/

NAMEC=  .*2
        .TXT    /TEMPC.TM/

NAMED=  .*2
        .TXT    /TEMPD.TM/

NAMEE=  .*2
        .TXT    /MTO:6/

NAMEF=  .*2
        .TXT    /MTO:7/

NAMEG=  .*2
        .TXT    /$TTI/

NAMEH=  .*2
        .TXT    /$TTO/

NAMEI=  .*2
        .TXT    /$LPT/

NAMEJ=  .*2
        .TXT    /POSTSCRIPT.DA/

NAMEK=  .*2
        .TXT    /ASSIGNA.DA/

NAMEL=  .*2
        .TXT    /ASSIGNB.DA/

NAMEM=  .*2
        .TXT    /ASSIGNC.DA/

ASGBF=  .*2
        .TXT    /01234  ABCDEFGHIJKLMNOP**<15>/

HSBBF=  .*2
.TXT/0123456789  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  <15>/

LVEBF=  .*2
        .TXT    /0123456789*    *    12345    ABCDEFGHIJKLMNOP**<15>/

HEAD1=  .*2
        .TXT    /<16><11><11><11>BDACS ONLINE PRINTOUT<12><15>/

HEAD2=  .*2
        .TXT    /<16><11><11><11>PREAMBLE & POSTSCRIPT FILES<12><15>/
```

```
HEAD3=  .*2
        .TXT    /<16><11><11><11>REDUCED MUX & LSB DATA<12><15>/

HEAD4=  .*2
        .TXT    / TIME(US)    STATE   SIGNAL  MNEMONIC<12><15>/

HEAD5=  .*2
        .TXT    /<16><11><11><11>MSB ASSIGNMENT & DATA<12><15>/

HEAD6=  .*2
        .TXT    / BIT#   MNEMONIC NAME<12><15>/

HEAD7=  .*2
.TXT/ TIME(NS)  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16<12><15>/
HEAD8=  .*2
        .TXT    /<16><11><11><11>METHOD FILE<12><15>/

MSG01=  .*2
        .TXT    /<12>POSTSCRIPT FILE OK? (Y,N) /
PAGE1=  .*2
        .TXT    /<11><11><11><11><11><11><11>PAGE00000<15>/

BKLN1=  .*2
        .TXT    /<15>/

BKLN2=  .*2
        .TXT    /<11><12><15>/

TOF=    .*2
        .TXT    /<14>/

UFBPT=  .*2
        .BLK    110

        BDACS           ;INITIAL MSB BIT #
        000405          ;CH # AND BLOCK COUNT
        177777          ;CURRENT BLOCK #
BUFRA=  .
        .BLK    400
        0

        BDACF           ;INITIAL MUX POINT #
        000404          ;CH # AND BLOCK COUNT
        177777          ;CURRENT BLOCK #
BUFRB=  .               ;BUFFER FOR ASSIGNB.DA
        .BLK    400
        0

        BDACQ           ;INITIAL LSB POINT #
        000405          ;CH # AND BLOCK COUNT
        177777          ;CURRENT BLOCK #
BUFRC=  .               ;BUFFER FOR ASSIGNC.DA
        .BLK    400
        0

BUFFR=  .               ;INPUT DATA BUFFER
        .BLK    BDAC9
        0

        .END
```

APPENDIX A

A-7. Overlay Module No. 6--ERMSG.SR

NAME BLOCK NAME= ERMSG.SR

TIME BLOCK

```
        .TITL   ERMSG   ;JCI    10 MAR 76

        .TXTM   1

        .ENT    OVST6

        .EXTD   OVRTN RECOV ERRTN ERCOD

        .NREL

BTMSK:  377             ;BYTE MASK
PHSNO:  MSG00+10

OVST6:  LDA     0,ERCOD ;ERROR CODE
        MOV #   0,0,SNR ;SKIP IF ERROR CONDITION
        JMP     NORML   ;NORMAL EXIT
        COM #   0,0,SNR ;SKIP IF NOT SYSTEM ERROR
        JMP     SYSTM   ;ABNORMAL EXIT
SMSG:   ADC     1,1     ;FORCE A -1
        STA     1,ERCOD ;RESET THE CODE
        LDA     1,BTMSK ;BYTE MASK
        AND     0,1     ;ERROR #
        SUBS    1,0     ;PHASE #
        LDA @   2,PHSNO ;WORD IN PHASE MASSAGE
        ADD     0,2     ;ADD IN OFFSET
        STA @   2,PHSNO ;RESTORE THE WORD
        LDA     2,TBLPT ;TABLE OF POINTERS
        ADD     0,2     ;ADD IN PHASE # OFFSET
        LDA     2,0,2   ;GET THE POINTER
        ADD     1,2     ;ADD IN ERROR # OFFSET
        LDA     2,0,2   ;MESSAGE POINTER
        MOVZL   2,2     ;MOVE REC. BIT TO CARRY, FORM BYTE POINTER
        SUBCL   1,1     ;MOV CARRY TO 1B15
        STA     1,ERCOD ;HOLD RECOVERABLE ERROR CODE
        STA     2,MSG   ;STORE THE POINTER
        JSR     TYPMG   ;TYPE THE MESSAGE
                MSG00*2
        JSR     TYPMG   ;TYPE THE MESSAGE
MSG:            0
        DSZ     ERCOD   ;SKIP IF RECOVERABLE ERROR
        JMP     .+2
        JMP @   RECOV   ;RETURN TO ROOT BINARY

NORML:  SUB     1,1,SKP ;SET FLAG 0
SYSTM:  ADC     1,1     ;SET FLAG -1
        .SYSTM
        .RESET          ;RESET ALL I/O CHANNELS
        HALT
        LDA     0,MTANM ;'MTO'
```

```
        .SYSTM
        .RLSE               ;RELEASE MTO
        JMP     .+1         ;IGNORE ERRORS
        LDA     2,RECOV     ;RETRIEVE ERROR CODE
        MOV #   1,1,SZR     ;SKIP IF NORMAL EXIT
        JMP     .+4
        .SYSTM
        .RTN                ;NORMAL RETURN TO CLI
        HALT
        .SYSTM
        .ERTN               ;ERROR RETURN TO CLI
        HALT

TYPMG:  LDA     0,0,3       ;GET BYTE POINTER
        SUBZL   2,2         ;FORCE +1 FOR CH#
        STA     3,USP       ;STORE RETURN ADDRESS
        .SYSTM
        .WRL    77          ;WRITE A LINE
        JMP @   ERRTN       ;ERROR RETURN
        JMP     1,3         ;NORMAL RETURN

MTANM:  .+1*2               ;MTO NAME POINTER
        .TXT    /MTO/

TBLPT:  .                   ;START OF POINTERS
        PHAS1
        PHAS2
        PHAS3
        PHAS4
        PHAS5

PHAS1:  .                   ;POINTER TO START OF PHAS1 ERRORS
        MSG11
        MSG12
        MSG13
        MSG14
        MSG15
        MSG16
        MSG17
        MSG18
        MSG19
        MSH10
        MSH11
        MSH12
        MSG99

PHAS2:  .                   ;POINTER TO START OF PHAS2 ERRORS
        MSG21
        MSG22+180
        MSG23
        MSG24
        MSG99

PHAS3:  .                   ;POINTER TO START OF PHAS3 ERRORS
        MSG23
        MSG99
```

APPENDIX A

```
PHAS4:  .                   ;POINTER TO START OF PHAS4 ERRORS
        MSG41
        MSG41
        MSG42
        MSG43
        MSG99

PHAS5:  .                   ;POINTER TO START OF PHAS5 ERRORS
        MSG51
        MSG52
        MSG53+1B0
        MSG54
        MSG99
MSG00=  .
        .TXT    /<12>ERROR IN PHASE  0<15>/

MSG11=  .
        .TXT    /MONITOR POINT UNDERFLOW<15>/
MSG12=  .
        .TXT    /MONITOR POINT OVERFLOW<15>/
MSG13=  .
        .TXT    /OUTPUT CONTROL TABLE OVERFLOW<15>/
MSG14=  .
        .TXT    /OUTPUT LIST NOT IN SEQUENTIAL ORDER<15>/
MSG15=  .
        .TXT    /OUTPUT LIST DELTA TIME OVERFLOW<15>/
MSG16=  .
        .TXT    /CONTROL POINT UNDERFLOW<15>/
MSG17=  .
        .TXT    /CONTROL LIST OVERFLOW<15>/
MSG18=  .
        .TXT    /SP-DP OVERFLOW<15>/
MSG19=  .
        .TXT    /EOL, NO NUMBER FOUND<15>/
MSH10=  .
        .TXT    /SAMPLE RATE OUT OF BOUNDS<15>/
MSH11=  .
        .TXT    /LOW SPEED BUFFER MONITOR POINT OUT OF BOUNDS<15>/
MSH12=  .
        .TXT    /HS BUFFER SAMPLE RATE OUT OF BOUNDS<15>/

MSG21=  .
        .TXT    /DISK ERROR<15>/
MSG22=  .
        .TXT    /DISK OVERFLOW<15>/
MSG23=  .
        .TXT    /MULTIPLEXER ERROR<15>/
MSG24=  .
        .TXT    /INPUT BUFFER OVERRUN<15>/

MSG41=  .
        .TXT    /CURRENT TIME OVERFLOW<15>/
MSG42=  .
        .TXT    /MAJOR SEQUENCE SLIP OCCURRED<15>/
MSG43=  .
        .TXT    /MAX SEQUENCE ERROR COUNT EXCEEDED<15>/
```

```
MSG51=  .
        .TXT    /ASSIGN FILE READ ERROR OCCURRED<15>/
MSG52=  .
        .TXT    /ASSIGN FILE SEQUENCE ERROR OCCURRED<15>/
MSG53=  .
        .TXT    /LINE COUNT ERROR<15>/
MSG54=  .
        .TXT    /MUX OR LSB POINT # OUT OF BOUNDS<15>/

MSG99=  .
        .TXT    /UNKNOWN ERROR<15>/

        .END
```

APPENDIX A

A-8. BDACS System Parameters--BDACS.SR

```
NAME BLOCK   NAME=  BDACS.SR

TIME BLOCK

        .TITL   BDACS   ;JCI    19 FEB 76

NOP=    000401          ;NOOP (JMP .+1)
CLK=    054             ;RTC DEVICE CODE (HARDWARE)
MUX=    032             ;MUX INTERFACE DEVICE CODE (HARDWARE)
MUXOB=  300             ;MUX OUTPUT  BUFFER ADDRESS (HARDWARE)
IMCLK=  000004          ;CLK INTERRUPT MASK BIT
IMMUX=  001000          ;MUX INTERRUPT MASK BIT
IMDKP=  000400          ;DKP INTERRUPT MASK BIT

BDACA=  200             ;MUX INPUT SIZE
BDACB=  100             ;MUX OUTPUT SIZE
BDACD=  144             ;STARTING CYLINDER # FOR RAW DATA STORE
BDACE=  144             ;MAX # OF CYLINDERS USED
BDACF=  1               ;MIN MUX INPUT POINT #
BDACG=  200             ;MAX MUX INPUT POINT #
BDACH=  401             ;MIN MUX OUTPUT POINT #
BDACI=  476             ;MAX MUX OUTPUT POINT #
BDACJ=  200             ;MAX # OUTPUT POINT CHANGES
BDACK=  03              ;LENGTH OF ENTRY IN OUTPUT CONTROL LIST
BDACL=  03              ;LENGTH OF ENTRY IN REDUCED DATA BUFFER
BDACM=  6000            ;LENGTH OF MUX DATA BLOCK
BDACN=  144             ;MAX ALLOWABLE SEQUENCE ERRORS
BDACO=  4000            ;HS BUFFER SIZE
BDACP=  200             ;LS BUFFER SIZE
BDACQ=  1               ;MIN LSB POINT #
BDACR=  200             ;MAX LSB POINT #
BDACS=  1               ;MIN HSB BIT #
BDACT=  20              ;MAX HSB BIT #
BDACU=  60              ;MIN MUX SAMPLE RATE
BDACV=  7777            ;MAX MUX SAMPLE RATE
BDACW=  764             ;MIN HSB SAMPLE RATE
BDACX=  23420           ;MAX HSB SAMPLE RATE

BDAC1=  BDACA/20        ;# INPUT WORDS
BDAC2=  BDACB/20        ;# OUTPUT WORDS
BDAC3=  BDACP/20        ;# LS BUFFER WORDS
BDAC5=  BDACJ*BDACL     ;LENGTH OF OUTPUT LIST BUFFER
BDAC6=  BDAC1+BDAC1     ;DISPLACEMENT FOR LPT MASKS
BDAC7=  BDAC1-1         ;# OF LS WORDS
BDAC8=  BDACM/2         ;# OF HS/LS PAIRS PER MUX BUFFER
BDAC9=  BDACL*400       ;REDUCED DATA BUFFER SIZE


END BLOCK
```

DISTRIBUTION

DEFENSE DOCUMENTATION CENTER
CAMERON STATION, BUILDING 5
ALEXANDRIA, VA 22314
ATTN DDC-TCA (12 COPIES)

COMMANDER
US ARMY MATERIEL DEVELOPMENT
& READINESS COMMAND
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333
ATTN DRXAM-TL, HQ TECH LIBRARY
ATTN DRCPP/MG C. M. MCKEEN, JR.
ATTN DRCPP-M/COL R. W. SPECKER
ATTN DRCPM-SCM-WF
ATTN DRCDE-D/MR. HUNT
ATTN DRCDE-D/COL J. F. BLEECKER
ATTN DRCDE, DIR FOR DEV & ENGR
ATTN DRCDE-DE/H. DARRACOTT
ATTN DRCMS-I/DR. R. P. UHLIG
ATTN DRCMS-I/MR. E. O'DONNEL
ATTN DRCDMD-ST/N. L. KLEIN

COMMANDER
US ARMY ARMAMENT MATERIEL
READINESS COMMAND
ROCK ISLAND ARSENAL
ROCK ISLAND, IL 61201
ATTN DRSAR-ASF, FUZE & MUNITION DIV
ATTN DRSAR-PDM/J. A. BRINKMAN
ATTN DRCPM-VFF

COMMANDER
USA MISSILE & MUNITIONS CENTER & SCHOOL
REDSTONE ARSENAL, AL 35809
ATTN ATSK-CTD-F

COMMANDING OFFICER
NAVAL TRAINING EQUIPMENT CENTER
ORLANDO, FL 32813
ATTN TECHNICAL LIBRARY

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD
ARLINGTON, VA 22209
ATTN TECH INFORMATION OFFICE
ATTN DIR, STRATEGIC TECHNOLOGY
ATTN DIR, TACTICAL TECHNOLOGY

DIRECTOR
DEFENSE COMMUNICATION ENG CENTER
1860 WIEHLE AVENUE
RESTON, VA 22090
ATTN R104, M. J. RAFFENSPERGER
ATTN R800, R. E. LYONS
ATTN R320, A. IZZO

DIRECTOR
DEFENSE INTELLIGENCE AGENCY
WASHINGTON, DC 20301
ATTN DI-2, WEAPONS & SYSTEMS DIV

DIRECTOR
DEFENSE NUCLEAR AGENCY
WASHINGTON, DC 20305
ATTN PETER HAAS, DEP DIR,
SCIENTIFIC TECHNOLOGY
ATTN RAEV, MAJ S. O. KENNEDY, SR.
ATTN VLIS, LTC ADAMS

DEPARTMENT OF DEFENSE
DIRECTOR OF DEFENSE RESEARCH & ENGINEERING
WASHINGTON, DC 20301
ATTN DEP DIR (TACTICAL WARFARE PROGRAMS)
ATTN DEP DIR (TEST & EVALUATION)
ATTN DEFENSE SCIENCE BOARD
ATTN ASST DIR SALT SUPPORT GP/
MR. J. BLAYLOCK

CHAIRMAN
JOINT CHIEFS OF STAFF
WASHINGTON, DC 20301
ATTN J-3, NUCLEAR WEAPONS BR
ATTN J-3, EXER PLANS & ANALYSIS DIV
ATTN J-5, NUCLEAR DIR NUCLEAR POLICY BR
ATTN J-5, REQUIREMENT & DEV BR
ATTN J-6, COMMUNICATIONS-ELECTRONICS

DEPARTMENT OF DEFENSE
JOINT CHIEFS OF STAFF
STUDIES ANALYSIS & GAMING AGENCY
WASHINGTON, DC 20301
ATTN STRATEGIC FORCES DIV
ATTN GEN PURPOSE FORCES DIV
ATTN TAC NUC BR
ATTN SYS SUPPORT BR

ASSISTANT SECRETARY OF DEFENSE
PROGRAM ANALYSIS AND EVALUATION
WASHINGTON, DC 20301
ATTN DEP ASST SECY (GEN PURPOSE PROG)
ATTN DEP ASST SECY (REGIONAL PROGRAMS)
ATTN DEP ASST SECY (RESOURCE ANALYSIS)

DEPARTMENT OF THE ARMY
OFFICE, SECRETARY OF THE ARMY
WASHINGTON, DC 20301
ATTN ASST SECRETARY OF THE ARMY (I&L)
ATTN DEP FOR MATERIEL ACQUISITION
ATTN ASST SECRETARY OF THE ARMY (R&D)

DEPARTMENT OF THE ARMY
ASSISTANT CHIEF OF STAFF FOR INTELLIGENCE
WASHINGTON, DC 20301
ATTN DAMI-OC/COL J. A. DODDS
ATTN DAMI-TA/COL F. M. GILBERT

US ARMY SECURITY AGENCY
ARLINGTON HALL STATION
4000 ARLINGTON BLVD
ARLINGTON, VA 22212
ATTN DEP CH OF STAFF RESEARCH & DEVELOPMENT

DISTRIBUTION (Cont'd)

DEPARTMENT OF THE ARMY
US ARMY CONCEPTS ANALYSIS AGENCY
8120 WOODMONT AVENUE
BETHESDA, MD 20014
ATTN COMPUTER SUPPORT DIV
ATTN WAR GAMING DIRECTORATE
ATTN METHODOLOGY AND RESOURCES DIRECTORATE
ATTN SYS INTEGRATION ANALYSIS DIRECTORATE
ATTN JOINT AND STRATEGIC FORCES DIRECTORATE
ATTN FORCE CONCEPTS AND DESIGN DIRECTORATE
ATTN OPERATIONAL TEST AND EVALUATION AGENCY

DIRECTOR
NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MD 20755

COMMANDER-IN-CHIEF
EUROPEAN COMMAND
APO NEW YORK, NY 09128

HEADQUARTERS
US EUROPEAN COMMAND
APO NEW YORK, NY 09055

DIRECTOR
WEAPONS SYSTEMS EVALUATION GROUP
OFFICE, SECRETARY OF DEFENSE
400 ARMY-NAVY DRIVE
WASHINGTON, DC 20305
ATTN DIR, LT GEN GLENN A. KENT

DEPARTMENT OF THE ARMY
DEPUTY CHIEF OF STAFF FOR OPERATIONS
& PLANS
WASHINGTON, DC 20301
ATTN DAMO-RQD/COL E. W. SHARP
ATTN DAMO-SSP/COL D. K. LYON
ATTN DAMO-SSN/LTC R. E. LEARD
ATTN DAMO-SSN/LTC B. C. ROBINSON
ATTN DAMO-RQZ/COL G. A. POLLIN, JR.
ATTN DAMO-TCZ/MG T. M. RIENZI
ATTN DAMO-ZD/A. GOLUB
ATTN DAMO-RQA/COL M. T. SPEIR

DEPARTMENT OF THE ARMY
CHIEF OF RESEARCH DEVELOPMENT
AND ACQUISITION OFFICE
WASHINGTON, DC 20301
ATTN DAMA-RAZ-A/R. J. TRAINOR
ATTN DAMA-CSM-N/LTC OGDEN
ATTN DAMA-WSA/COL W. E. CROUCH, JR.
ATTN DAMA-WSW/COL L. R. BAUMANN
ATTN DAMA-CSC/COL H. C. JELINEK
ATTN DAMA-CSM/COL H. R. BAILEY
ATTN DAMA-WSZ-A/MG D. R. KEITH
ATTN DAMA-WSM/COL J. B. OBLINGER, JR.
ATTN DAMA-PPR/COL D. E. KENNEY

COMMANDER
BALLISTIC MISSILE DEFENSE SYSTEMS
P.O. BOX 1500
HUNTSVILLE, AL 35807
ATTN BMDSC-TEN/MR. JOHN VEFNEMAN

COMMANDER
US ARMY FOREIGN SCIENCE
AND TECHNOLOGY CENTER
220 SEVENTH ST., NE
CHARLOTTESVILLE, VA 22901

DIRECTOR
US ARMY MATERIEL SYSTEMS ANALYSES ACTIVITY
ABERDEEN PROVING GROUND, MD 21005
ATTN DRXSY-C/DON R. BARTHEL
ATTN DRXSY-T/P. REID

COMMANDER
US ARMY SATELLITE COMMUNICATIONS AGENCY
FT. MONMOUTH, NJ 07703
ATTN LTC HOSMER

DIRECTOR
BALLISTIC RESEARCH LABORATORIES
ABERDEEN PROVING GROUND, MD 21005
ATTN DRXBR-XA/MR. J. MESZAROS

COMMANDER
US ARMY AVIATION SYSTEMS COMMAND
12TH AND SPRUCE STREETS
ST. LOUIS, MO 63160
ATTN DRCPM-AAH/ROBERT HUBBARD

DIRECTOR
EUSTIS DIRECTORATE
US ARMY AIR MOBILITY R&D LABORATORY
FORT EUSTIS, VA 23604
ATTN SAVDL-EU-MOS/MR. S. POCILUYKO
ATTN SAVDL-EU-TAS(TETRACORE)

COMMANDER
2D BDE, 101ST ABN DIV (AASLT)
FORT CAMPBELL, KY 42223
ATTN AFZB-KB-SO
ATTN DIV SIGNAL OFFICER,
AFBZ-SO/MAJ MASON

COMMANDER
US ARMY ELECTRONICS COMMAND
FT. MONMOUTH, NJ 07703
ATTN PM, ATACS/DRCPM-ATC/LTC DOBBINS
ATTN DRCPM-ATC-TM
ATTN PM, ARTADS/DRCPM-TDS/BG A. CRAWFORD
ATTN DRCPM-TDS-TF/COL D. EMERSON
ATTN DRCPM-TDS-TO
ATTN DRCPM-TDS-FB/LTC A. KIRKPATRICK

US ARMY ELECTRONICS COMMAND (Cont'd)
ATTN PM, MALOR/DRCPM-MALR/COL W. HARRISON
ATTN PM, NAVCOM/DRCPM-NC/COL C. MCDOWELL, JR.
ATTN PM, REMBASS/DRCPM-RBS/COL R. COTTEY, SR.
ATTN DRSEL-TL-IR/MR. R. FREIBERG
ATTN DRSEL-SA/NORMAN MILLSTEIN
ATTN DRSEL-MA-C/J. REAVIS
ATTN DRSEL-CE-ES/J. A. ALLEN

COMMANDER
US ARMY MISSILE MATERIEL READINESS COMMAND
REDSTONE ARSENAL, AL 35809
ATTN DRSMI-FRR/DR. F. GIPSON
ATTN DRCPM-HA/COL P. RODDY
ATTN DRCPM-LCCX/L. B. SEGGEL (LANCE)
ATTN DRCPM-MD/GENE ASHLEY (PATRIOT)
ATTN DRCPM-MP
ATTN DRCPM-PE/COL SKEMP (PERSHING)
ATTN DRCPM-SHO
ATTN DRCPM-TO
ATTN DRSMI-R, RDE & MSL DIRECTORATE

COMMANDER
US ARMY ARMAMENT RESEARCH
& DEVELOPMENT COMMAND
DOVER NJ 07801
ATTN DRDAR-ND-V/DANIEL WAXLER

COMMANDER
US ARMY TANK/AUTOMOTIVE MATERIEL
READINESS COMMAND
WARREN, MI 48090
ATTN DRSI-RHT/MR. P. HASEK
ATTN DRCPM(XM-L)/MR. L. WOOLCOT
ATTN DRCPM-GCM-SW/MR. R. SLAUGHTER

PRESIDENT
DA, HA, US ARMY ARMOR AND ENGINEER BOARD
FORT KNOX, KY 40121
ATTN STEBB-MO/MAJ SANZOTERRA

COMMANDER
WHITE SANDS MISSILE RANGE
WHITE SANDS MISSILE RANGE, NM 88002
ATTN STEWS-TE-NT/MARVIN SQUIRES

COMMANDER
TRASANA
SYSTEM ANALYSIS ACTIVITY
WHITE SANDS, NM 88002
ATTN ATAA-TDO/DR. D. COLLIER

COMMANDER
197TH INFANTRY BRIGADE
FORT BENNING, GA 31905
ATTN COL WASIAK

COMMANDER
US ARMY COMMUNICATIONS COMMAND
FORT HUACHUCA, AZ 85613
ATTN ACC-AD-C/H. LASITTER (EMP STUDY GP)

COMMANDER
USA COMBINED ARMS COMBAT DEVELOPMENTS
ACTIVITY
FT. LEAVENWORTH, KS 66027
ATTN ATCAC
ATTN ATCACO-SD/LTC L. PACHA
ATTN ATCA/COC/COL HUBBERT
ATTN ATCA-CCM-F/LTC BECKER
ATTN ATSW-TD-3 NUCLEAR STUDY
TEAM/LT D. WILKINS

PROJECT MANAGER
MOBILE ELECTRIC POWER
7500 BACKLICK ROAD
SPRINGFIELD, VA 22150
ATTN DRCPM-MEP

DEPUTY COMMANDER
US ARMY NUCLEAR AGENCY
7500 BACKLICK RD
BUILDING 2073
SPRINGFIELD, VA 22150
ATTN MONA-WE/COL A. DEVERILL

COMMANDER
US ARMY SIGNAL SCHOOL
FT. GORDON, GA 30905
ATTN AISO-CID/BILL MANNELL
ATTN ATST-CTD-CS/CAPT G. ALEXANDER
(INTACS)
ATTN ATSO-CID-CS/MR. TAYLOR
ATTN ATSN-CD-OR/MAJ CARR

DIRECTOR
JOINT TACTICAL COMMUNICATIONS OFFICE
FT. MONMOUTH, NJ 07703
ATTN TRI-TAC/NORM BECHTOLD

COMMANDER
US ARMY COMMAND AND GENERAL STAFF COLLEGE
FORT LEAVENWORTH, KS 66027

COMMANDER
US ARMY COMBAT DEVELOPMENTS EXPERIMENTATION
COMMAND
FORT ORD, CA 93941

COMMANDER
HQ MASSTER
FORT HOOD, TX 76544

COMMANDER
US ARMY AIR DEFENSE SCHOOL
FORT BLISS, TX 79916
ATTN ATSA-CD

COMMANDER
US ARMY ARMOR SCHOOL
FORT KNOX, KY 40121
ATTN ATSB-CTD (2 COPIES)

DISTRIBUTION (Cont'd)

COMMANDER
US ARMY AVIATION CENTER
FORT RUCKER, AL 36360
ATTN ATST-D-MS (2 COPIES)

COMMANDER
US ARMY ORDNANCE CENTER AND SCHOOL
ABERDEEN PROVING GROUND, MD 21005
ATTN USAOC&S
ATTN ATSL-CTD

COMMANDER
US ARMY SIGNAL SCHOOL
FORT GORDON, GA 30905
ATTN ATSS-CTD (2 COPIES)

COMMANDER
US ARMY ENGINEER SCHOOL
FORT BELVOIR, VA 22060
ATTN ATSE-CTD (2 COPIES)

COMMANDER
US ARMY INFANTRY SCHOOL
FORT BENNING, GA 31905
ATTN ATSH-CTD (2 COPIES)

COMMANDER
US ARMY INTELLIGENCE CENTER AND SCHOOL
FORT HUACHUCA, AZ 85613
ATTN ATSI-CTD (2 COPIES)

COMMANDER
US ARMY FIELD ARTILLERY SCHOOL
FORT SILL, OK 73503
ATTN ATSF-CTD (2 COPIES)

CHIEF OF NAVAL OPERATIONS
NAVY DEPARTMENT
WASHINGTON, DC 20350
ATTN NOP-932, SYS EFFECTIVENESS DIV
CAPT E. V. LANEY
ATTN NOP-9860, COMMUNICATIONS BR
COR L. LAYMAN
ATTN NOP-351, SURFACE WEAPONS BR
CAPT G. A. MITCHELL
ATTN NOP-622C, ASST FOR NUCLEAR
VULNERABILITY, R. PIACESI

COMMANDER
NAVAL ELECTRONICS SYSTEMS COMMAND, HQ
2511 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VA 20360
ATTN PME-117-21, SANGUINE DIV

HEADQUARTERS, NAVAL MATERIEL COMMAND
STRATEGIC SYSTEMS PROJECTS OFFICE
1931 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VA 20390
ATTN NSP2201, LAUNCHING & HANDLING
BRANCH, BR ENGINEER, P. R. FAUROT
ATTN NSP-230, FIRE CONTROL & GUIDANCE
BRANCH, BR ENGINEER, D. GOLD
ATTN NSP-2701, MISSILE BRANCH,
BR ENGINEER, J. W. PITSENBERGER

COMMANDER
NAVAL SURFACE WEAPONS CENTER
WHITE OAK, MD 20910
ATTN CODE 222, ELECTRONICS & ELECTRO-
MAGNETICS DIV
ATTN CODE 431, ADVANCED ENGR DIV

US AIR FORCE, HEADQUARTERS
DCS, RESEARCH & DEVELOPMENT
WASHINGTON, DC 20330
ATTN DIR OF OPERATIONAL REQUIREMENTS
AND DEVELOPMENT PLANS, S/V &
LTC P. T. DUESBERRY

COMMANDER
AF WEAPONS LABORATORY, AFSC
KIRTLAND AFB, NM 87117
ATTN ES, ELECTRONICS DIVISION
ATTN EL, J. DARRAH
ATTN TECHNICAL LIBARY
ATTN D. I. LAWRY

COMMANDER
AERONAUTICAL SYSTEMS DIVISION, AFSC
WRIGHT-PATTERSON AFB, OH 45433
ATTN ASD/YH, DEPUTY FOR B-1

COMMANDER
HQ SPACE AND MISSILE SYSTEMS ORGANIZATION
P.O. 96960 WORLDWAYS POSTAL CENTER
LOS ANGELES, CA 90009
ATTN S7H, DEFENSE SYSTEMS APL SPO
ATTN XRT, STRATEGIC SYSTEMS DIV
ATTN SYS, SURVIVABILITY OFC

SPACE AND MISSILE SYSTEMS ORGANIZATION
NORTON AFB, CA 92409
ATTN MMH, HARD ROCK SILO DEVELOPMENT

COMMANDER
AF SPECIAL WEAPONS CENTER, AFSC
KIRTLAND AFB, NM 87117

DISTRIBUTION (Cont'd)

ASSISTANT CHIEF OF STAFF FOR
COMMUNICATIONS ELECTRONICS
XVIII AIRBORNE CORPS
FORT BRAGG, NC 28307
ATTN AFZA-CE/LTC K. KILLINGSTEAD

HARRY DIAMOND LABORATORIES
ATTN RAMSDEN, JOHN J., COL, COMMANDER/
FLYER, I.N./LANDIS, P.E./
SOMMER, H./OSWALD, R. B.
ATTN CARTER, W.W., DR., TECHNICAL
DIRECTOR/MARCUS, S.M.
ATTN KIMMEL, S., PAO
ATTN CHIEF, 0021
ATTN CHIEF, 0022
ATTN CHIEF, LAB 100
ATTN CHIEF, LAB 200
ATTN CHIEF, LAB 300
ATTN CHIEF, LAB 400
ATTN CHIEF, LAB 500
ATTN CHIEF, LAB 600
ATTN CHIEF, DIV 700
ATTN CHIEF, DIV 800
ATTN CHIEF, LAB 900
ATTN CHIEF, LAB 1000
ATTN RECORD COPY, BR 041
ATTN HDL LIBRARY (5 COPIES)
ATTN CHAIRMAN, EDITORIAL COMMITTEE
ATTN CHIEF, 047
ATTN TECH REPORTS, 013
ATTN PATENT LAW BRANCH, 071
ATTN GIDEP OFFICE, 741
ATTN LANHAM, C., 0021
ATTN CHIEF, 0024
ATTN CHIEF, 1010
ATTN CHIEF, 1020 (20 COPIES)
ATTN CHIEF, 1030
ATTN CHIEF, 1040
ATTN CHIEF, 1050
ATTN NOON, T. V., 1020